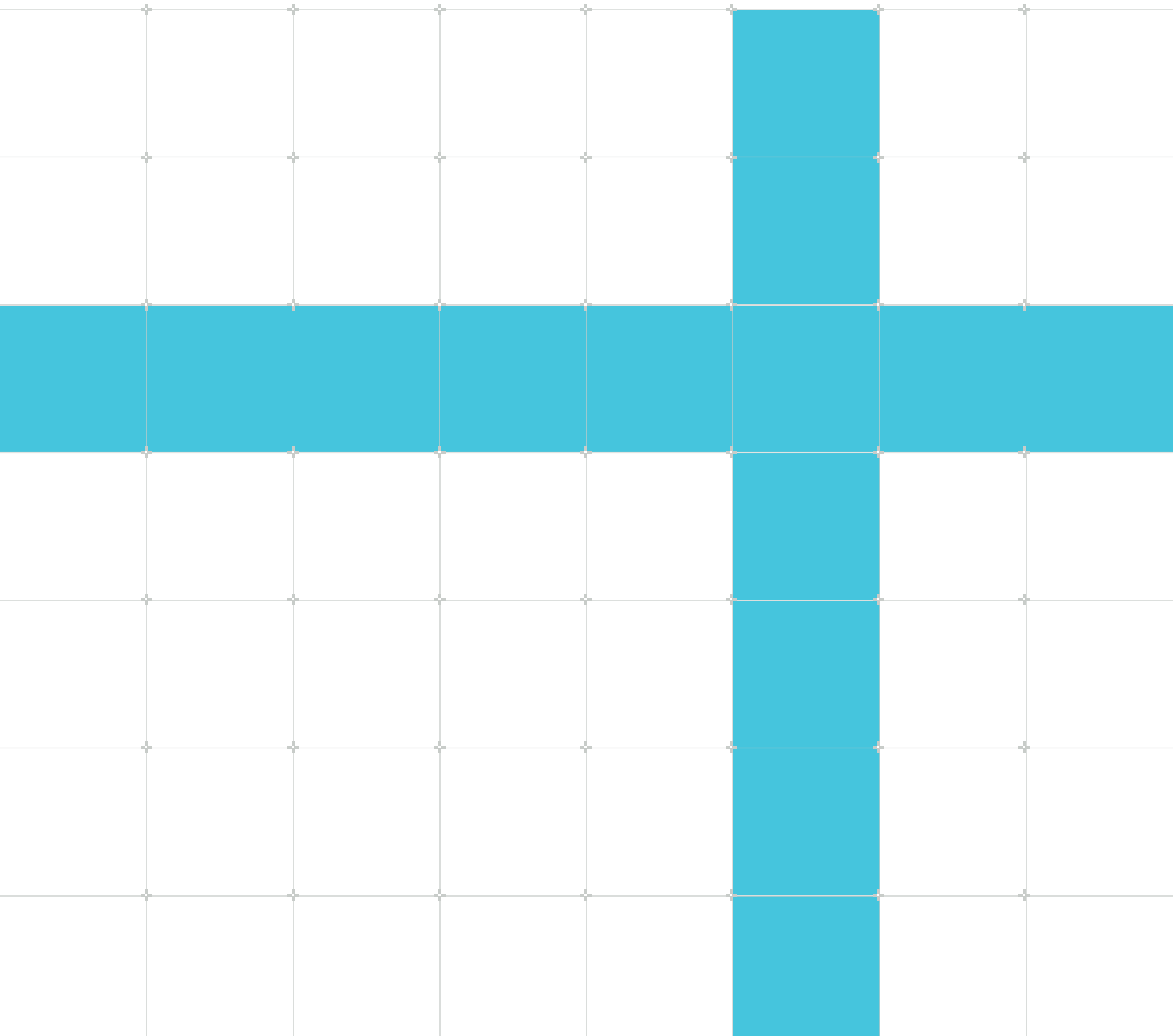
Arm® Cortex®-X3 Core

Revision: r1p2

Technical Reference Manual

Issue 07
101593_0102_07_en



Arm® Cortex®-X3 Core

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	14 August 2020	Confidential	First Alpha release for r0p0
0000-03	11 November 2020	Confidential	First Beta release for r0p0
0000-04	19 February 2021	Confidential	First limited access release for r0p0
0100-05	22 July 2021	Confidential	First early access release for r1p0
0101-06	28 June 2022	Non-Confidential	First early access release for r1p1
0102-07	16 December 2022	Non-Confidential	First release for r1p2

Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Feedback

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

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

Inclusive language commitment

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

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

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

Contents

1. Introduction

1.1 Product revision status

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

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

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

1.2 Intended audience

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

1.3 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

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

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

Convention	Use
italic	Citations.
bold	Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`

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

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

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

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

An important piece of information that needs your attention.

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

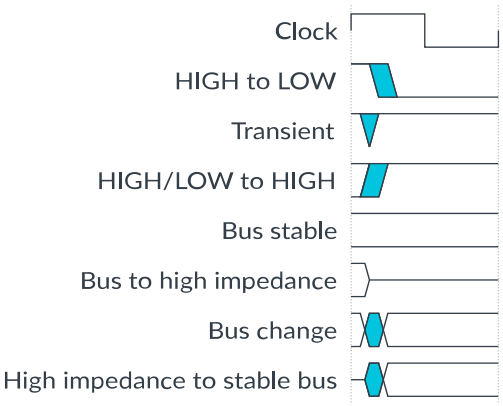
A reminder of something important that relates to the information you are reading.

Timing diagrams

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

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

Figure 1-1: Key to timing diagram conventions



Signals

The signal conventions are:

Signal level

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

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

Lowercase n

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

1.4 Useful resources

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

Access to Arm documents depends on their confidentiality:

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

Arm product resources	Document ID	Confidentiality
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	Non-Confidential
Arm® Cortex®-X3 Core Configuration and Integration Manual	101594	Confidential
Arm® Cortex®-X3 Core Cryptographic Extension Technical Reference Manual	101592	Non-Confidential
Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual	101382	Confidential
Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual	101381	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® 5 CHI Architecture Specification	IHI 0050	Non-Confidential
Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM) for Armv8-A	DDI 0598	Non-Confidential
Arm® Architecture Reference Manual Supplement The Scalable Vector Extension (SVE) for Armv8-A	DDI 0584	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	Non-Confidential
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	Non-Confidential
Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile	DDI 0587	Non-Confidential

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

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

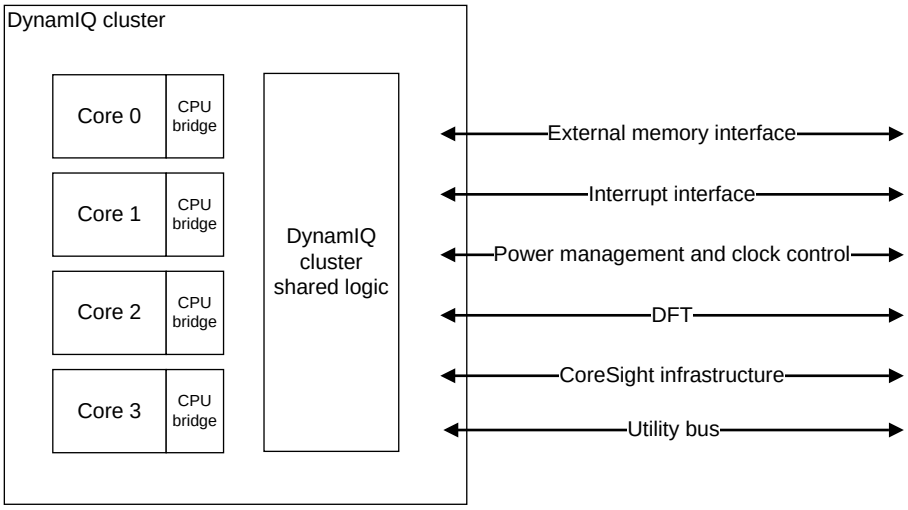
2. The Cortex®-X3 core

The Cortex®-X3 core is a high-performance and low-power product that 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 Cortex®-X3 core targets large-screen compute applications.

The Cortex®-X3 core is implemented inside a DynamIQ™-110 cluster and is always connected to the *DynamIQ™ Shared Unit-110* (DSU-110) that behaves as a full interconnect with L3 cache and snoop control. This configuration is also used in systems with different types of cores where Cortex®-X3 is the high-performance core.

The following figure shows an example configuration with four Cortex®-X3 cores in a DynamIQ™ cluster.

Figure 2-1: Cortex®-X3 example configuration



This manual applies to the Cortex®-X3 core only. Read this manual together with the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for detailed information about the DSU-110.

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

2.1 Cortex®-X3 core features

The Cortex®-X3 core can be used in a standalone DynamIQ™ configuration, which is a homogenous cluster of one to four Cortex®-X3 cores. It might also be used as the high-performance core in a heterogenous cluster.

However, regardless of the cluster configuration, the Cortex®-X3 core always has the same features.

Core features

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

Related information

3. Technical overview on page 37

2.2 Cortex®-X3 core configuration options

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

The Cortex®-X3 core implementation options include:

Vector datapath
: You can configure the Vector datapath to be 2x128b or 4x128b.

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

L2 Data RAM ECC granule
: You can configure the L2 Data RAM ECC granule to be 128b or 256b.

L2 cache size
: You can configure the L2 cache to be 512KB or 1MB. The cores in the cluster can have different cache sizes.

L2 transaction queue size
: You can configure the L2 transaction queue size to be 72, 80, 88, or 96.

PMU Event Counters
: Configure the number of PMU events counters to be 6 or 20.

CoreSight 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 AMBA® *Trace Bus* (ATB) FIFO to be 4, 8, 16, 32, or 64.

Timing closure
: You can configure the L2 data cache RAMs timing behavior. For more information, see *Cortex-X3 configuration parameters* in the *Arm® Cortex®-X3 Core Configuration and Integration Manual*.

For detailed configuration options and guidelines, see *RTL configuration process* in the *Arm® Cortex®-X3 Core Configuration and Integration Manual*.

2.3 DSU-110 dependent features

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

The following table describes which DSU-110 dependent features are supported in your Cortex®-X3 core.

Table 2-1: Cortex®-X3 core features that have a dependency on the DSU-110

Feature	Supported in the Cortex®-X3 core	Dependency on the DSU-110
Direct connect	No	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.
SMCRYPTODISABLE signal supported	Yes	Affects the external signals of the DSU-110. For the connection information of this signal, see the *DynamIQ™ Shared Unit-110 signals* section in the *Functional integration* chapter of the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual*.
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.
DISPBLKy 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	40-bit	Affects the CHI and AXI master port bus widths. For more details, see the following parts of the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*: • *CHI master interface* • *AXI master interface*

2.4 Supported standards and specifications

The Cortex®-X3 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 Cortex®-X3 core supports AArch64 at all Exception levels, EL0 to EL3, and supports all mandatory features of each architecture version.

The following tables show, for each Armv8-A architecture version, the optional features that the Cortex®-X3 core supports.

Table 2-2: Armv8.0-A optional feature support in the Cortex®-X3 core

Feature	Status	Notes
Cryptographic Extension	Supported using a configurable option	For more information, see the *Arm® Cortex®-X3 Core Cryptographic Extension Technical Reference Manual*. This extension is licensed separately and access to the documentation is restricted by contract with Arm.
FEAT_AdvSIMD, Advanced *Single Instruction Multiple Data* (SIMD) Extension	Supported	For more information, see 13. Advanced SIMD and floating-point support on page 100.
FEAT_FP, Floating-point Extension	Supported	
FEAT_PMUv3, Performance Monitors Extension	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on this feature.
FEAT_DGH, Data Gathering Hint	Supported	Adds the Data Gathering Hint instruction to the hint space.

Table 2-3: Arm®v8.1-A optional feature support in the Cortex®-X3 core

Feature	Status	Notes
FEAT_HAFDBS, Hardware Management of the Access Flag and Dirty State	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on these features.
FEAT_VMID16, 16-bit *Virtual Machine IDentifier* (VMID)	Supported	
FEAT_PAN3, Support for SCTLR_ELx.EPAN	Supported	

Table 2-4: Arm®v8.2-A optional feature support in the Cortex®-X3 core

Feature	Status	Notes
FEAT_HPDS2, Translation Table Page-Based Hardware Attributes	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on these features.
FEAT_PCSRv8p2, PC Sample-based Profiling	Supported	
FEAT_SHA512, Advanced SIMD SHA512 instructions	Supported as part of Armv8-A Cryptographic Extension	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on these features.
FEAT_SHA3, Advanced SIMD SHA3 instructions		

Feature	Status	Notes
FEAT_SM3, Advanced SIMD SM3 instructions		
FEAT_SM4, Advanced SIMD SM4 instructions		
FEAT_BF16, 16-bit floating-point instructions	Supported	
FEAT_I8MM, Int8 Matrix Multiply instructions	Supported	
FEAT_MPAM, *Memory System Resource Partitioning and Monitoring* (MPAM) Extension	Supported	See the *Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A* for information on this extension.
FEAT_SVE, Scalable Vector Extension	Supported	See 14. Scalable Vector Extensions support on page 101 and the *Arm® Architecture Reference Manual for A-profile architecture* for information on this extension.
FEAT_LPA, Large *Physical Address* (PA) and *Intermediate PA* (IPA) Support	Not supported	-
FEAT_LVA, Large *Virtual Address*(VA) Support	Not supported	-
FEAT_LSMAOC, Load/Store Multiple Atomicity and Ordering Controls	Not supported	-
FEAT_AA32HPD, AArch32 Hierarchical Permission Disables	Not supported	-
FEAT_SPE, Statistical Profiling Extension	Supported	For more information on this extension, see 21. Statistical Profiling Extension support on page 148 and the *Arm® Architecture Reference Manual for A-profile architecture*.

Table 2-5: Arm®v8.3-A optional feature support in the Cortex®-X3 core

Feature	Status	Notes
FEAT_NV, Nested Virtualization	Not supported	-
FEAT_CCIDX, Extended Cache Index	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on this feature.
FEAT_Pauth2, Pointer Authentication enhancements	Supported	-
FEAT_FPAC	Supported	-

Table 2-6: Arm®v8.4-A optional feature support in the Cortex®-X3 core

Feature	Status	Notes
FEAT_AMUv1	Supported	See the *Arm® Architecture Reference Manual for A-profile architecture* for information on this feature.
FEAT_NV2, enhanced support for Nested Virtualization	Not supported	-

Table 2-7: Arm®v8.5-A optional feature support in the Cortex®-X3 core

Feature	Status	Notes
FEAT_MTE and FEAT_MTE2, Memory Tagging Extension	Supported	The Cortex®-X3 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.
FEAT_MTE3, MTE Asymmetric Fault Handling	Supported	MTE enhancement.
FEAT_RNG, Random Number Generator instructions	Not supported	-
FEAT_ExS, Context Synchronization and Exception Handling	Not supported	-

The following table shows the Arm®v9.0-A features that the Cortex®-X3 core supports.

Table 2-8: Arm®v9.0-A feature support in the Cortex®-X3 core

Feature	Status	Notes
FEAT_SVE2, Scalable Vector Extension 2	Supported	See 14. Scalable Vector Extensions support on page 101.
FEAT_SVE_AES, Scalable Vector AES Instructions	Supported	See the *Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension* for more information.
FEAT_SVE_BitPerm, Scalable Vector Bit Permutes	Supported	
FEAT_SVE_PMULL, Scalable Vector Polynomial Multiply Instructions which Generate a 128-bit Result	Supported	
FEAT_SVE_SHA3, Scalable Vector SHA3 Instructions	Supported	
FEAT_SVE_SM4, Scalable Vector SM Instructions	Supported	
FEAT_ETE, *Embedded Trace Extension* (ETE)	Supported	See 18. Embedded Trace Extension support on page 129.
FEAT_TRBE, *TRace Buffer Extension* (TRBE)	Supported	See 19. Trace Buffer Extension support on page 141.
FEAT_TME, *Transactional Memory Extension* (TME)	Not supported	-

The following table shows the other standards and specifications that the Cortex®-X3 core supports.

Table 2-9: Other standards and specifications support in the Cortex®-X3 core

Standard or specification	Version	Notes
FEAT_GICv4p1, *Generic Interrupt Controller* (GIC) 4.1	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.
FEAT_RASv1p1, *Reliability, Availability, and Serviceability* (RAS) Extensions	v1.1	All extensions up to Arm®v9.0-A with *Error Correcting Code* (ECC) configured. See 11. RAS Extension support on page 92 for more information on the implementation of this extension in the core.

Standard or specification	Version	Notes
CoreSight	v3.0	See the *Arm® CoreSight™ Architecture Specification v3.0* for more information.
FEAT_ECBHB, *Exploitative Control using Branch History Buffer* information between exception levels	-	The branch history information created in a context before an exception to a higher exception level, using AArch64, cannot be used by code before that exception. This prevents exploitative control of the execution of any indirect branches in code in a different context after the exception.

Related information

3.1 Core components on page 37

2.5 Test features

The Cortex®-X3 core provides test signals that enable the use of both *Automatic Test Pattern Generation* (ATPG) and *Memory Built-In Self Test* (MBIST) to test the core logic and memory arrays.

The Cortex®-X3 core includes an *Automatic Test Pattern Generation* (ATPG) test interface that provides signals to control the *Design For Test* (DFT) features of the core. To prevent problems with DFT implementation, carefully consider usage of these signals.

Arm also provides *Memory Built-In Self Test* (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 MBIST interfaces instead of the supplied Arm interfaces.

See the *Design for Test integration guidelines* chapter of the *Arm® Cortex®-X3 Core Configuration and Integration Manual* for the list of test signals and information on their usage. Also see the *Design for Test integration guidelines* chapter of the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for the list of external scan control signals.

2.6 Design tasks

The Cortex®-X3 core is delivered as a synthesizable RTL description in SystemVerilog HDL. Before you can use the Cortex®-X3 core, you must implement it, integrate it, and program it. Implementation and integration choices affect the behavior and features of the core.

A different party can perform each of the following tasks:

Implementation

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

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

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

Integration

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

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

Software programming

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

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

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

See the *RTL configuration process* chapter of the *Arm® Cortex®-X3 Core Configuration and Integration Manual* and *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for implementation options. See also the *Functional integration* chapter of the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for signal descriptions.

2.7 Product revisions

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

Table 2-10: Product revisions

Revision	Notes
r0p0	First release
r1p0	First early access release for r1p0 includes updated ERXPFGCTL_EL1 register description, minor changes to IMP_CPUECTLR_EL1 and CTR_EL0 bit descriptions tables and the modified introduction to Debug chapter.
r1p1	First early access release for r1p1 includes updated powerup and powerdown sequence, description on FEAT_ECBHB and updated write streaming mode
r1p2	Changes to system ID registers throughout document.

Changes in functionality that have an impact on the documentation also appear in C.1 Revisions on page 1496.

3. Technical overview

All components in the Cortex®-X3 core are always present. These components are designed to make the Cortex®-X3 core a high-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 unit and trace buffer
- The *Performance Monitoring Unit* (PMU)
- The *Activity Monitoring Unit* (AMU)
- The *Generic Interrupt Controller* (GIC) CPU interface

The Cortex®-X3 core interfaces with the *DynamIQ™ Shared Unit-110* (DSU-110) through the CPU bridge.

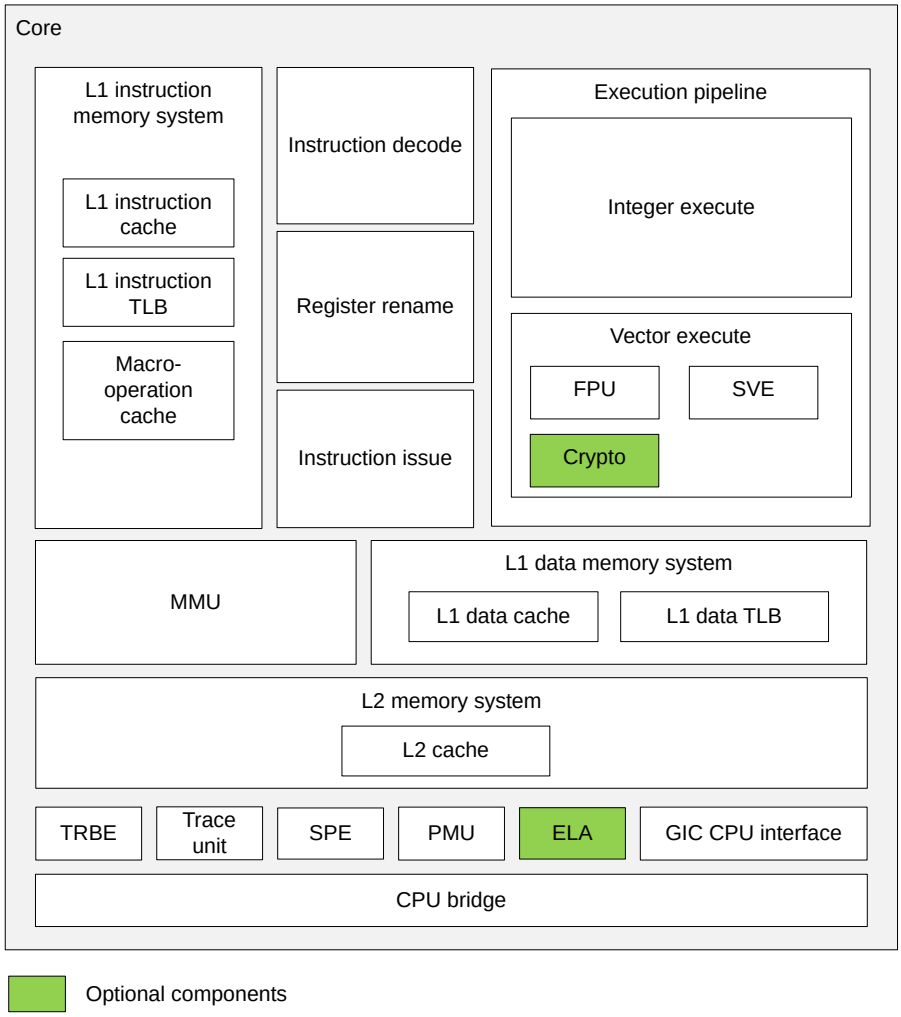
The Cortex®-X3 core implements the Arm®v9.0-A architecture and supports all previous Armv8-A architectures up to Arm®v8.5-A . The programmers model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in 2.4 Supported standards and specifications on page 31.

3.1 Core components

The Cortex®-X3 core includes components designed to make it a high-performance and low-power product. The Cortex®-X3 core includes a CPU bridge that connects the core to the *DynamIQ™ Shared Unit-110* (DSU-110). The DSU-110 connects the core to an external memory system and the rest of the SoC.

The following figure shows the Cortex®-X3 core components.

Figure 3-1: Cortex®-X3 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 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 Cortex®-X3 cores. The Cryptographic Extension adds new instructions to the Advanced SIMD and the *Scalable Vector Extension* (SVE) instruction sets that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption.
- The *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-2, SHA-3, SHA-224, SHA-256, SHA-384, and SHA-512.
    - The SVE2 versions of the SHA-3 instructions EOR3, XAR, and BCAX are supported even when CRYPTO support is not configured.
- Armv8.2-SM SM3 hash function and SM4 encryption and decryption instructions.
- Finite field arithmetic that is used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography.


Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension under an additional license to the Cortex®-X3 core license.

Scalable Vector Extension

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

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 1MB. The L2 memory system is connected to the DSU-110 through an asynchronous CPU bridge.

Embedded Trace Extension and Trace Buffer Extension

The Cortex®-X3 core supports a range of debug, test, and trace options including a trace unit and a trace buffer.

The Cortex®-X3 core also includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the Cortex®-X3 core are described in this manual. For more information about the *Embedded Logic Analyzer* (ELA), see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

Statistical Profiling Extension

The Cortex®-X3 core implements the *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.

Performance Monitoring Unit

The *Performance Monitoring Unit* (PMU) provides 6 or 20 performance monitors (depending on your configuration) that can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

Activity Monitoring Unit

The Cortex®-X3 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitors in the *Activity Monitoring Unit* (AMU) provide useful information for system power management and persistent monitoring.

GIC CPU interface

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

CPU bridge

In a cluster, there is one CPU bridge between each Cortex®-X3 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 57
7. L1 instruction memory system on page 65
8. L1 data memory system on page 68
9. L2 memory system on page 73
12. GIC CPU interface on page 97
13. Advanced SIMD and floating-point support on page 100
17. Performance Monitors Extension support on page 115
18. Embedded Trace Extension support on page 129

3.2 Interfaces

The *DynamIQ™ Shared Unit-110* (DSU-110) manages all Cortex®-X3 core external interfaces to the *System on Chip* (SoC).

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

3.3 Programmer's model

The Cortex®-X3 core implements the Arm®v9.0-A architecture and supports all Armv8-A architectures up to Arm®v8.5-A. The Cortex®-X3 core supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

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

Related information

2.4 Supported standards and specifications on page 31

4. Clocks and resets

The Cortex®-X3 core supports hierarchical clock gating to provide dynamic power savings. The core also supports Warm and Cold resets.

Each Cortex®-X3 core has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the CPU bridge.

In addition, the Cortex®-X3 core implements extensive clock gating that includes:

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

The Cortex®-X3 core receives the following reset signals from the *DynamIQ™ Shared Unit-110* (DSU-110) side of the CPU bridge:

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

See the *Clocks and resets* and *Power and reset control with Power Policy Units* chapters of 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 Cortex®-X3 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

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

The static power management includes the following features:

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

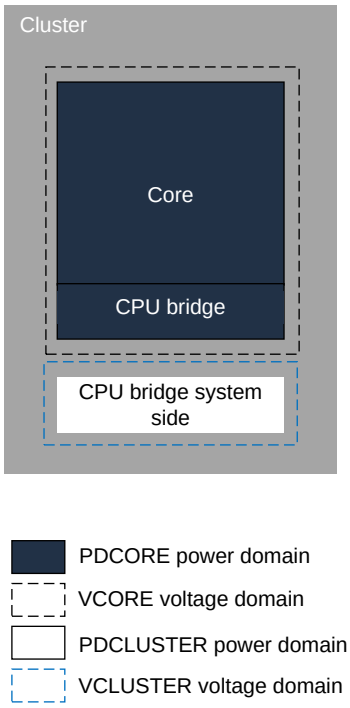
5.1 Voltage and power domains

The *DynamIQ™ Shared Unit-110* (DSU-110) *Power Policy Units* (PPUs) control power management for the Cortex®-X3 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 Cortex®-X3 core logic and part of the core asynchronous bridge that belongs to the VCORE domain. The PDCLUSTER power domain contains the part of the CPU bridge that belongs to the VCLUSTER domain.

The following figure shows the Cortex®-X3 core power domain and voltage domain. It also shows the cluster power domain and voltage domain that cover the system side of the CPU bridge.

Figure 5-1: Cortex®-X3 core voltage and power domains



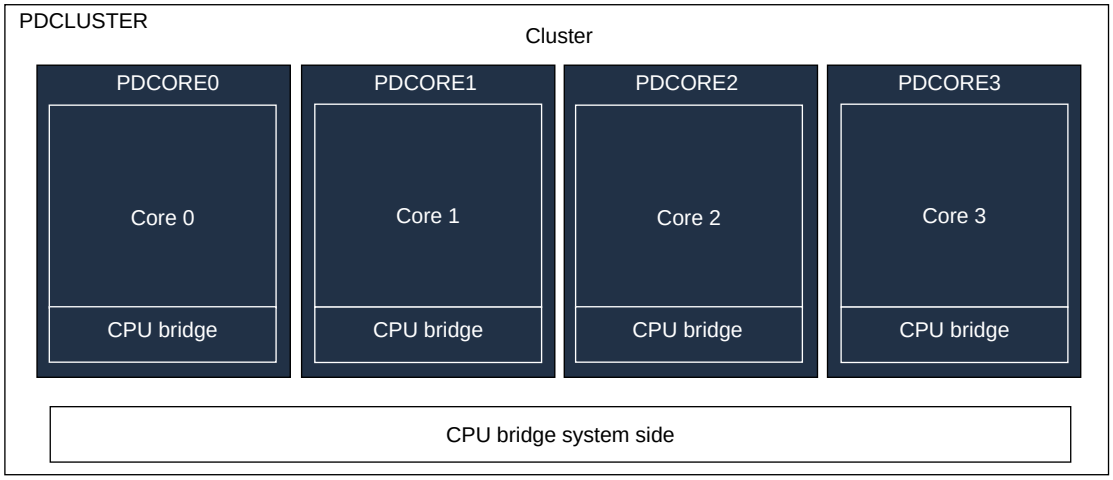
You can tie the VCORE and VCLUSTER voltage domains to the same supply if one of the following is true:

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

In a cluster with multiple Cortex®-X3 cores, there is one PDCORE<n> power domain per core, where n is the core instance number. If a core is not present, then the corresponding power domain is not present.

The following figure shows an example of the power domains with four Cortex®-X3 cores in a cluster.

Figure 5-2: Core power domains in a cluster with four Cortex®-X3 cores



Clamping cells between power domains are inferred through power intent files rather than instantiated in the RTL. See the *Power management* chapter of the *Arm® Cortex®-X3 Core Configuration and Integration Manual* for more information.

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

5.2 Architectural clock gating modes

The *Wait For Interrupt* (WFI) and *Wait For Event* (WFE) instructions put the core into a low-power mode. These instructions work by architecturally disabling the clock at the top of the clock tree. The core remains fully powered and retains all 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.

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

When the core executes the `WFI` 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` instruction also ensures 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 occurs:

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

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

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

5.2.2 Low-power state behavior considerations

You must consider how certain events affect the *Wait for Interrupt* (WFI) and *Wait for Event* (WFE) low-power state behavior of the Cortex®-X3 core.

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

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


Note

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

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

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

5.3 Power control

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

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

The PPUs decide and request any change in power mode. The Cortex®-X3 core then performs any actions necessary to reach the requested power mode. For example, the core might gate clocks, clean caches, or disable coherency before it accepts the request.

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

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

5.4 Core power modes

The Cortex®-X3 core power domain has a defined set of power modes and corresponding legal transitions between these modes. The power mode of each core can be independent of other cores in a cluster.

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

The following table shows the supported Cortex®-X3 power modes.

Power modes that are not shown in the following table are not supported and must not occur.

Table 5-1: Cortex®-X3 core power modes

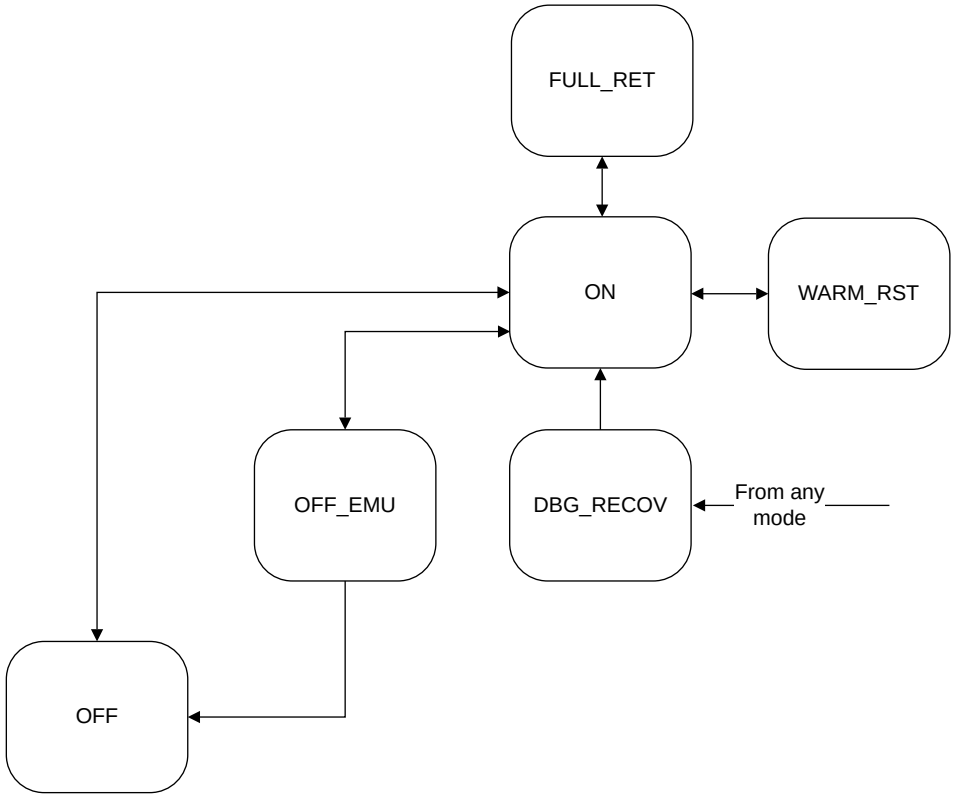
Power mode	Short name	Power state
On	ON	The core is powered up and active.
Full retention	FULL_RET	The core is in retention. In this mode, only power that is required to retain register and RAM state is available. The core is not operational. A core must be in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state before it enters this mode.
Off	OFF	The core is powered down.

Power mode	Short name	Power state
Emulated Off	OFF_EMU	Emulated off mode permits you to debug the powerup and powerdown cycle without changing the software. In this mode, the core powerdown is normal, except: • The clock is not gated and power is not removed when the core is powered down. • Only the 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.

Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management and powerup and powerdown sequences described in 5.6 Cortex-X3 core powerup and powerdown sequence on page 54.

The following figure shows the supported modes for the Cortex®-X3 core power domain and the legal transitions between them.

Figure 5-3: Cortex®-X3 core power mode transitions



Related information

5.2 Architectural clock gating modes on page 46
5.2.1 Wait for Interrupt and Wait for Event on page 46
5.4.4 Full retention mode on page 51

5.4.1 On mode

In On power mode, the Cortex®-X3 core is on and fully operational.

The core can be initialized into On mode. When a transition to On mode completes, 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 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 Off.

5.4.4 Full retention mode

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

The core can enter into Full retention mode when all of the following conditions are met:

- The retention timer has expired.
- The core is in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state.
- The core clock is not temporarily enabled for L1 or L2 snoops, cache, or *Translation Lookaside Buffer* (TLB) maintenance operations, or debug or *Generic Interrupt Controller* (GIC) access.

The core can exit Full retention mode when it detects any of the following:

- A WFI or WFE wakeup event, as defined in the *Arm® Architecture Reference Manual for A-profile architecture*.
- An event that requires the core clock to be temporarily enabled without exiting the WFI or WFE low-power state. For example, an L1 or L2 snoop, a cache or TLB maintenance operation, a debug access on the debug APB bus, or a GIC access.

Related information

5.2.1 Wait for Interrupt and Wait for Event on page 46

5.4.5 Debug recovery mode

Debug recovery mode can be used to assist debug of external watchdog-triggered reset events.

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

Debug recovery also supports preserving the *Reliability, Availability, and Serviceability* (RAS) state, in addition to the cache contents. In this case, a transition to debug recovery is made from any state.

When in Debug recovery mode, a cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to On mode.

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

This 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, 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 a system deadlock could result.

If the system sends a snoop to the cluster during this mode, then depending on the cluster state, the snoop might get a response and disturb the contents of the caches, or it 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 Cortex®-X3 core when the core receives a Warm reset signal from the *DynamIQ™ Shared Unit-110* (DSU-110) side of the CPU bridge:

The Cortex®-X3 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 for A-profile architecture* for more information about RMR_EL3.

5.5 Performance and power management

The Cortex®-X3 core implements *Performance and Power Management* (PPM) features that can be used to limit high activity events within the core, or trade off efficiency versus peak performance.

The PPM features are:

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

5.5.1 Maximum Power Mitigation Mechanism

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

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

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

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

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

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

Related information

B.2.1 CPUPPMCR, Power Performance Management Register on page 980

5.5.2 Performance Defined Power

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

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

The PDP has an impact on:

- Core power reduction. The core power is reduced and the efficiency is increased.
- External memory system power reduction. Memory request bandwidth is modulated to reduce power in the memory system.

5.5.3 Dispatch block

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

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

5.6 Cortex®-X3 core powerup and powerdown sequence

There is no specific sequence to power up the Cortex®-X3 core. To power down the core, you must follow a specific sequence. There are no software steps required to bring a core into coherence after reset.

To powerdown the Cortex®-X3 core:

1. If required, save the state of the core to system memory to allow for retrieval of the core state during core powerup.
2. Disable interrupts to the core.
   a. Disable the interrupt enable bits in the ICC_IGRPEN0_EL1 and ICC_IGRPEN1_EL1 registers.
   b. Set the GIC distributor wake-up request for the core using the GICR_WAKER register.
   c. Read the GICR_WAKER register to confirm that the ChildrenAsleep bit indicates that the interface is quiescent.
3. Disable the interrupt outputs from the RAS registers. Alternatively, re-direct the core RAS fault and error interrupt outputs to the system error manager. For more information, see 5.6.1 Managing RAS fault and error interrupts during the core powerdown on page 54.
4. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
5. Execute an `ISB` instruction.
6. Execute a `WFI` instruction. Once the `WFI` instruction is executed, the powerdown sequence cannot be interrupted.

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

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

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

5.6.1 Managing RAS fault and error interrupts during the core powerdown

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

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

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

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

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

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

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

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

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

If an uncorrectable error occurs during the powerdown, the core remains powered ON and the software remains inactive. The system error manager is then responsible for resetting the entire cluster and the wider system that interacts with the core and cluster. To use this approach, the system must be designed to allow the core RAS error interrupt to re-route to the system error manager. As the core RAS registers are only accessible to software running on the core, the system error manager is unable to identify where the uncorrectable error occurred within the core.

5.7 Debug over powerdown

The Cortex®-X3 core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This behavior enables debug to continue through powerdown scenarios, rather than having to re-establish a connection each time the core is powered up.

The debug over powerdown logic is part of the DebugBlock in the *DynamIQ™ Shared Unit-110* (DSU-110). The DebugBlock is external to the cluster, and must remain powered on during the debug over powerdown process.

See the *Debug* chapter of the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for more information.

6. Memory management

The *Memory Management Unit* (MMU) is responsible for translating an input address to an output address. This translation is based on address mapping and memory attribute information that is available in the Cortex®-X3 core internal registers and translation tables. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

An address translation from an input address to an output address is described as a stage of address translation. The Cortex®-X3 core can perform:

- Stage 1 translations that translate an input *Virtual Address* (VA) to an output *Physical Address* (PA) or *Intermediate Physical Address* (IPA).
- Stage 2 translations that translate an input IPA to an output PA.
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The Cortex®-X3 core performs translation table walks for each stage of the translation.

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

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

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

6.1 Memory Management Unit components

The Cortex®-X3 *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 Cortex®-X3 core implements a two-level TLB structure.

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

The following table describes the MMU components.

Table 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 • 48 entries
L1 *TRace Buffer Extension* (TRBE) TLB	• VA to PA translations of any page and block size • 1 entry
L2 TLB	• Shared by instructions and data • VA to PA mappings for 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, and 1GB block sizes • *Intermediate Physical Address* (IPA) to PA mappings for: ◦ 2MB and 1GB block sizes in a 4KB translation granule ◦ 32MB block size in a 16KB translation granule ◦ 512MB block size in a 64KB granule • Intermediate (descriptor) PAs obtained during a translation table walk • 8-way set associative • 2048 entries
Translation table prefetcher	• Detects access to contiguous translation tables and prefetches the next one • Can be disabled in the ECTLR register

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

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

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

6.2 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), and a set of memory properties that includes type and access permissions.

Each TLB entry is associated with either a particular *Address Space Identifier* (ASID) or 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 60

6.3 Translation Lookaside Buffer match process

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

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

- EL3 in Secure state
- EL2, or EL0 in *Virtualization Host Extensions* (VHE) mode, in secure state and Non-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:

- Its VA, moderated by the page size such as 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 *Address Space Identifier* (ASID) matches the current ASID held in the TTBR0_ELx or TTBR1_ELx register associated with the target translation regime, or the entry is marked global.
- The *Virtual Machine Identifier* (VMID) matches the current VMID held in the VTTBR_EL2 register.

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

- EL2 in secure state and Non-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 Cortex®-X3 core generates a memory access, the *Memory Management Unit* (MMU) searches for the requested *Virtual Address* (VA) in the *Translation Lookaside Buffers* (TLBs). If it is not present, then it is a miss and the MMU proceeds by looking up the translation table during a translation table walk.

When the Cortex®-X3 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.

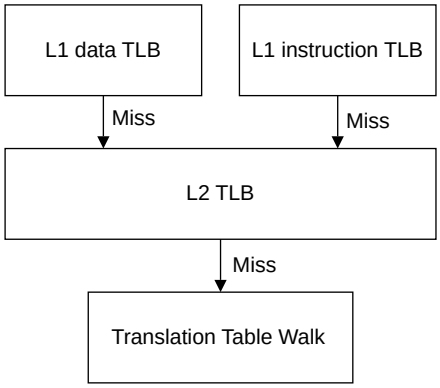
Address translation is performed only when the MMU is enabled. They can also be disabled for a particular translation base register, in which case the MMU returns a translation fault.

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 for A-profile architecture* 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 65
8. L1 data memory system on page 68
9. L2 memory system on page 73

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 and VTCR_EL2. Translation table descriptors include the *Dirty Bit Modifier* (DBM) field to support the hardware management of dirty state.

The Cortex®-X3 core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the Cortex®-X3 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 the 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 acquire semantics, stores to any memory type, and AtomicSt, cache maintenance, TLBI, and IC instructions.

Cortex®-X3 takes a synchronous abort on a Normal memory ldrx that receives an non-EXOK response from CHI. The abort is asynchronous for Device memory ldrx. For strx, OK and EXOK responses are expected and do not cause aborts. NDErr and DErr responses for WriteNoSnp Excl=1 cause asynchronous aborts.

Misprogramming contiguous hints

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, then the Cortex®-X3 core does not generate a translation fault.

Conflict aborts

The Cortex®-X3 core does not generate Conflict aborts.

6.7 Memory behavior and supported memory types

The Cortex®-X3 core supports memory types defined in the Armv8-A architecture.

Device memory types have the following attributes:

G – Gathering

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

R – Reordering

The capability to reorder transactions

E – Early Write Acknowledgement

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


Note

In the following table, n denotes a "non-" prefix, that is, n means the capability is not allowed.

The following table shows how memory types are supported in the Cortex®-X3 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 63.	Write-Back Cacheable (any allocation hint)	Write-Back Cacheable No Allocate	Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the *DynamIQ™ Shared Unit-110* (DSU-110) is 0 (No Allocate)
Normal	See Table 6-3: Shareability for Normal memory on page 63.	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-Through are treated as Non-cacheable, and therefore are Outer Shareable.

6.7.1 Page-based hardware attributes

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

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

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

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

PBHA combination between stage 1 and stage 2 on memory accesses

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

When stage 1 and stage 2 are enabled:

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

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

Mismatched aliases

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

7. L1 instruction memory system

The Cortex®-X3 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	**L1 I-cache** Pseudo-*Least Recently Used* (LRU) cache replacement policy for L1 **L0 MOP-cache** *Not Recently Used* (NRU) 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 57.

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.

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 51

7.2 L1 instruction cache Speculative memory accesses

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

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

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

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

7.3 Program flow prediction

The Cortex®-X3 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 instructions. Program flow prediction hardware predicts all branch instructions, and includes:

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

Exception return branch instructions are not predicted.

Return stack

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

In AArch64, any of the following instructions causes a return stack push:

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

Any of the following instructions cause a return stack pop:

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

The following instructions are not predicted:

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

8. L1 data memory system

The Cortex®-X3 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	*Re-reference Interval Prediction* (RRIP) replacement scheme
Interface with integer execute pipeline and vector execute	• 4×64-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 instruction memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in 6. Memory management on page 57.

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.

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.

To maintain data coherency between multiple cores, the Cortex®-X3 core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

Related information

5.4.5 Debug recovery mode on page 51

8.2 Instruction implementation in the L1 data memory system

The Cortex®-X3 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 Cortex®-X3 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 Cortex®-X3 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 Cortex®-X3 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, in the A64 instruction set, any instruction that has a mnemonic starting with LDX, LDAX, STX, or STLX.

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

8.4 Data prefetching

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

Preload instructions

The Cortex®-X3 core supports the AArch64 prefetch memory instructions, PRFM.

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

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

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 for A-profile architecture*.

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 IMP_CPUECTLR_EL1 register controls the prefetcher. For more information, see A.1.10 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 173.

Data cache zero

In the Cortex®-X3 core, the *Data Cache Zero by Virtual Address* (DC ZVA) instruction enables a block of 64 bytes in memory, aligned to 64 bytes in size, to be set to zero.

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

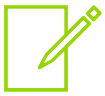
8.5 Write streaming mode

The Cortex®-X3 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 memory system can detect when the core has written a full cache line before the linefill completes. If this situation is detected on a configurable number of consecutive linefills, then it switches into write streaming mode.

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


Note

More than the specified number of linefills might be observed on the master interface, before the memory system 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 Cortex®-X3 core has switched to write streaming mode, the memory system continues to monitor the bus traffic. It signals to the L2 or L3 cache to go into write streaming mode when it observes a further number of full cache line writes.

The write streaming threshold defines the number of consecutive cache lines that are fully written without being read before store operations stop causing cache allocations. You can configure the write streaming threshold for each cache:

- IMP_CPUECTLR_EL1.L2_WR_THR configures the L2 write streaming mode threshold.
- IMP_CPUECTLR_EL1.L3_WR_THR configures the L3 write streaming mode threshold.
- IMP_CPUECTLR_EL1.L4_WR_THR configures the L4 write streaming mode threshold.
- IMP_CPUECTLR_EL1.DRAM_WR_THR configures the DRAM write streaming mode threshold.

Related information

A.1.10 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 173

9. L2 memory system

The Cortex®-X3 L2 memory system connects the core with the *DynamIQ™ Shared Unit-110* (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 Cortex®-X3 core in a cluster.

The following table shows the L2 memory system features.

Table 9-1: L2 memory system features

Feature	Type
L2 cache	512KB or 1MB
	8-way set associative, 4 banks
	Physically Indexed, Physically Tagged (PIPT)
	Always protected with *Error Correcting Code* (ECC)
Cache line length	64 bytes
Cache policy	Dynamic biased cache replacement policy
Interface with the 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 will 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 51

9.2 Support for memory types

The Cortex®-X3 core simplifies coherency logic by downgrading some memory types.

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

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

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

The additional attribute hints are used as follows:

Allocation hint

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

Transient hint

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

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

9.3 Transaction capabilities

The CHI Issue E interface between the Cortex®-X3 L2 memory system and the *DynamIQ™ Shared Unit-110* (DSU-110) provides transaction capabilities for the core.

The following table shows the maximum possible values for read, write, *Distributed Virtual Memory* (DVM) issuing, and snoop capabilities of the Cortex®-X3 L2 cache. There is a maximum limit of 92 outstanding transactions overall.

Table 9-2: Cortex®-X3 transaction capabilities

Attribute	Maximum value	Description
Write issuing capability	68, 76, 84, or 92	This is the maximum number of outstanding write transactions. It depends on the configured Transaction Queue (TQ) size: 72, 80, 88, or 96.
Read issuing capability	68, 76, 84, or 92	This is the maximum number of outstanding read transactions. It depends on the configured TQ size: 72, 80, 88, or 96.
Snoop acceptance capability	41, 45, 49, or 53	This is the maximum number of outstanding snoops accepted. It depends on the configured TQ size: 72, 80, 88, or 96.
DVM issuing capability	68, 76, 84, or 92	This is the maximum number of outstanding DVM operation transactions. It depends on the configured TQ size: 72, 80, 88, or 96.

10. Direct access to internal memory

The Cortex®-X3 core provides a mechanism to read the internal memory that the L1 and L2 caches and *Translation Lookaside Buffer* (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 that contain the contents of the internal memory. The internal memory is accessed by executing the **IMPLEMENTATION DEFINED** RAMINDEX instruction. The following table shows the registers that are used to read the data.

Table 10-1: System registers that contain results of reading 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: Cortex®-X3 L1 instruction cache tag location encoding

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

Bit field of Xn	Description
[17:14]	Reserved
[13:6]	Virtual address [13:6]
[5:0]	Reserved

Table 10-3: Cortex®-X3 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: Cortex®-X3 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 10-5: Cortex®-X3 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 10-6: Cortex®-X3 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]

Bit field of Xn	Description
[5:0]	Reserved

Table 10-7: Cortex®-X3 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 10-8: Cortex®-X3 L1 data TLB location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x0A`
[23:6]	Reserved
[5:0]	TLB Entry (0-47)

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 10-9: L1 instruction cache tag format for Instruction Register 0

Bit field	Description
[63:32]	Reserved
[31]	Non-secure identifier for the physical address
[30:3]	Physical address [39:12]
[2:1]	Instruction state [1:0] **`0b00`** Invalid **`0b01`** *Reserved for Future Use*(RFU) **`0b10`** RFU **`0b11`** Valid
[0]	Parity

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

Bit field	Description
[63:0]	Reserved

Table 10-11: L1 instruction cache tag format for Instruction Register 2

Bit field	Description
[63:0]	Reserved

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

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

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

Bit field	Description
[63:20]	0
[19:0]	Data [83:64]

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

Bit field	Description
[63:0]	0

10.1.3 L1 instruction TLB returned data

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

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

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

Bit field	Description
[63]	Virtual address [12]
[62:59]	PBHA [3:0]
[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]	Outer-shared
[50]	Inner-shared
[49:42]	0
[41:40]	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 10-16: L1 instruction TLB format for Instruction Register 1

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

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

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

10.1.4 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 10-18: L0 MOP cache format for Instruction Register 0

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

Table 10-19: L0 MOP cache format for Instruction Register 1

Bit field	Description
[63:28]	0
[27:0]	Macro-operation data [103:64]

Table 10-20: L0 MOP cache format for Instruction Register 2

Bit field	Description
[63:0]	0

10.1.5 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 10-21: L1 data cache tag format for Data Register 0

Bit field	Description
[63:57]	ECC
[56:28]	Non-secure identifier, physical address [39:12]
[27:25]	Reserved
[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 state
[1:0]	MESI: **0b00** Invalid **0b01** Shared **0b10** Exclusive **0b11** Modified

Table 10-22: L1 data cache tag format for Data Register 1

Bit field	Description
[63:0]	0

Table 10-23: L1 data cache tag format for Data Register 2

Bit field	Description
[63:0]	0

10.1.6 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 10-24: L1 data cache data format for Data Register 0

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

Table 10-25: L1 data cache data format for Data Register 1

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

Table 10-26: 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

10.1.7 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 10-27: 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 *Dirty Bit Modifier* (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 [1:0]
[32:17]	ASID [15:0]
[16:1]	VMID [15:0]
[0]	Valid

Table 10-28: L1 data TLB format for Data Register 1

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

Table 10-29: L1 data TLB format for Data Register 2

Bit field	Description
[5]	Tagged MTE
[4]	FWB override
[3]	PBHA [3]
[2]	PBHA [2]
[1]	PBHA [1]
[0]	PBHA [0]

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 10-30: Cortex®-X3 L2 cache tag location encoding for 512KB [2]

Bit field of Xn	Description
[31:24]	RAMID = `0x10`
[23:21]	Reserved
[20:18]	Way (0-7)
[17:16]	Reserved
[15:13]	Index [15:13]
[12:10]	XOR(Index [12:10], Way[2:0])

[2] Index[15:8]=XOR(physical address[15:8], physical address[23:16]) Index[7:6]=XOR(physical address[7:6], physical address[11:10])

Bit field of Xn	Description
[9:8]	Index [9:8]
[7:6]	XOR(Index [7:6], Index [11:10], Way[1:0])
[5:0]	Reserved

Table 10-31: Cortex®-X3 L2 cache tag location encoding for 1MB [3]

Bit field of Xn	Description
[31:24]	RAMID = `0x10`
[23:21]	Reserved
[20:19]	Way (0-7)
[18:17]	Reserved
[16:12]	Index [16:12]
[11:9]	XOR(Index [11:9], Way[2:0])
[8]	Index [8]
[7:6]	XOR(Index [7:6], Index [11:10], Way[2:1])
[5:0]	Reserved

Table 10-32: Cortex®-X3 L2 cache data location encoding for 512KB

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:21]	Reserved
[20:18]	Way (0-7)
[17:16]	Reserved
[15:13]	Index [15:13]
[12:10]	XOR(Index [12:10], Way[2:0])
[9:8]	Index [9:8]
[7:6]	XOR(Index [7:6], Index [11:10], Way[1:0])
[5:4]	Physical address [5:4]
[3:0]	Reserved

Table 10-33: Cortex®-X3 L2 cache data location encoding for 1MB

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:21]	Reserved
[20:19]	Way (0-7)
[18:17]	Reserved
[16:12]	XOR(Index [16:12], 5'b01000)
[11:9]	XOR(Index [11:9], Way[2:0])
[8]	Index [8]

[3] Index[16:8]=XOR(physical address[16:8], physical address[25:17]), Index[7:6]=XOR(physical physical address[7:6], physical address[11:10])

Bit field of Xn	Description
[7:6]	XOR(Index [7:6], Index [11:10], Way[2:1])
[5:4]	Physical address [5:4]
[3:0]	Reserved

Table 10-34: Cortex®-X3 L2 TLB location encoding

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

Table 10-35: Cortex®-X3 L2 victim location encoding

Bit field of Rd	Description
[31:24]	RAMID = `0x12`
[23:17]	Reserved
[16]	INV(Index[16])
[15:8]	Index [15:8]
[7:6]	XOR(Index[11:10], Index[7:6])
[5:0]	Reserved

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 tag cache format for instruction registers. In the first table:

For 512KB L2 cache

n=34, m=16

For 1MB L2 cache

n=33, m=17

Table 10-36: L2 tag cache format for Data Register 0

Bit field	Description
[63:n+22]	0
[n+21:n+13]	ECC
[n+12]	MPAM_PMG
[n+11:n+6]	MPAM_PARTID
[n+5]	MPAM_NS
[n+4:n+1]	PBHA[3:0]
[n:11]	Physical tag [39:m]

Bit field	Description
[10]	Non-secure
[9]	Reserved
[8:7]	Virtual address [13:12]
[6]	Shareable
[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 10-37: L2 tag cache format for Data Register 1

Bit field	Description
[63:0]	0

Table 10-38: 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 10-39: L2 data RAM format for Data Register 0

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

Table 10-40: L2 data RAM format for Data Register 1

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

Table 10-41: L2 data RAM format for Data Register 2 when L2 ECC granule equals 128

Bit field	Description
[63:15]	0
[14:6]	ECC[8:0] for Data [127:0]
[5:4]	Poison[0] is for Data [63:0] and Poison[1:0] is for Data [127:0]
[3:0]	MTE tags

Table 10-42: L2 data RAM format for Data Register 2 when L2 ECC granule equals 256

Bit field	Description
[63:11]	0
[10:6]	ECC[4:0] for Data [127:0][4]
[5:4]	Poison[0] is for Data [63:0] and Poison[1:0] is for Data [127:0]
[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 10-43: L2 TLB format for Instruction Register 0

Bit field	Description
[63]	Outer shareable
[62]	Inner shareable
[61]	Outer allocate

[4] ECC[9:5] for Data[255:128] are contained in the read of the next quad-word.

Bit field	Description
[60:58]	Memory attributes: **0b000** Device nGnRnE **0b001** Device nGnRE **0b010** Device nGRE **0b011** Device GRE **0b100** Non-cacheable **0b101** Write-Back No-Allocate **0b110** Write-Back Transient **0b111** Write-Back Read-Allocate and Write-Allocate
[57:54]	Reserved
[53:20]	Physical address When bit[6] is 0: • [53:26] = PA[39:12] • [25:20] = Reserved When bit[6] is 1: • [53:28] = PA[39:14] • [27:26] = PA[13:12] for page 3 (highest memory address) • [25:24] = PA[13:12]for page 2 • [23:22] = PA[13:12]for page 1 • [21:20] = PA[13:12]for page 0 (lowest memory address)

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

Table 10-44: L2 TLB format for Instruction Register 1

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

Table 10-45: L2 TLB format for Instruction Register 2

Bit field	Description
[63:22]	Reserved

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

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 10-46: Cortex®-X3 L2 victim format for data register 0

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

Table 10-47: Cortex®-X3 L2 victim format for data register 1

Bit field of Rd	Description
[63:0]	0

Table 10-48: Cortex®-X3 L2 victim format for data register 2

Bit field of Rd	Description
[63:0]	0

11. RAS Extension support

The Cortex®-X3 core supports the *Reliability, Availability, and Serviceability* (RAS) Extension, including all extensions up to Arm®v9.0-A.

In particular, the Cortex®-X3 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 cache tag and data RAMs, the L2 cache 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 cache 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 Cortex®-X3 core features the following nodes:

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

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

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

11.1 Cache protection behavior

The configuration of the *Reliability, Availability, and Serviceability* (RAS) Extension that is implemented in the Cortex®-X3 core includes cache protection. In this case, the Cortex®-X3 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Cortex®-X3 core have the following capability:

SEC parity

Single Error Correct. One bit of parity is applicable to the protected data. Errors are corrected by refetching cached data.

SECDED ECC

Single Error Correct, Double Error Detect. The data size is specific for each RAM and depends on the protection granule.

The following table shows which protection type is applied to each RAM in the Cortex®-X3 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
L0 *Macro-operation* (MOP) cache data	SEC parity
L1 instruction cache data	SEC parity
L1 instruction cache tag	SEC parity
L1 data cache data	SECDED ECC
L1 data cache tag	SECDED ECC
MMU Translation Cache (MMUTC)	SEC 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 SEC, 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 Cortex®-X3 core supports error containment for data errors, which means that detected errors are not silently propagated.

Data errors are propagated using data poisoning to ensure that a consumer is aware of the error. Uncorrectable L1 data cache and L2 cache tag errors are not containable.

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

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

11.3 Fault detection and reporting

When the Cortex®-X3 core detects a fault, it raises a *Fault Handling Interrupt* (FHI) exception or an *Error Recovery Interrupt* (ERI) exception through the fault or the error signals. FHIs and ERIs are reflected in the *Reliability, Availability, and Serviceability* (RAS) registers, which are updated in the node that detects the errors.

Fault handling interrupts

When ERR*n*CTLR.FI is set, all detected Deferred errors, Uncorrected errors, and overflows of the corrected error counters cause an FHI to be generated. When ERR*n*CTLR.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 ERR*n*CTLR.UI is set, all detected Uncorrected errors that are not deferred generate an ERI.

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

Related information

A.13 AArch64 RAS registers summary on page 898
A.13.4 ERXCTLR_EL1, Selected Error Record Control Register on page 906
A.13.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register on page 910

11.4 Error detection and reporting

When the Cortex®-X3 core consumes an error, it raises different exceptions depending on the error type.

The Cortex®-X3 core might raise:

- A *Synchronous External Abort* (SEA)

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

11.4.1 Error reporting and performance monitoring

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

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

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

Related information

17.1 Performance monitors events on page 115

11.5 Error injection

Error injection consists of inserting an error in the error detection logic to verify the reporting and recording structure.

Error injection uses the error detection and reporting registers to insert errors. The Cortex®-X3 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 ERXPFGCDN register. The value of the counter decrements on a per clock cycle basis.

For more information on the ERXPFGCDN register, see the *Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile*.


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

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

Table 11-2: RAS registers summary

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

12. GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC connects to the Cortex®-X3 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 DynamIQ™ cluster has a GIC CPU interface, which connects to a common external distributor component.

The GICv4.1 architecture implemented in the Cortex®-X3 core supports:

- Two security states
- *Secure Virtualization Extension* (SVE)
- *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
- Wake-up 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 Cortex®-X3 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 other GIC architectures to drive the nFIQ and nIRQ interrupt signals. If the Cortex®-X3 core is not integrated with an external GIC interrupt distributor component in the system (minimum GICv3 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 nVIFQ 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 the *Functional integration* chapter of the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for more information on these signals.

12.2 AArch64 GIC system register summary

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

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

Table 12-1: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Interrupt Priority Mask Register
ICV_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Virtual Interrupt Priority Mask Register
ICC_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Interrupt Acknowledge Register 0
ICV_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 0
ICC_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller End Of Interrupt Register 0
ICV_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller Virtual End Of Interrupt Register 0
ICC_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 0
ICV_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
ICV_BPR0_EL1	3	0	C12	C8	3	–	64-bit	Interrupt Controller Virtual Binary Point Register 0
ICC_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICC_DIR_EL1	3	0	C12	C11	1	–	64-bit	Interrupt Controller Deactivate Interrupt Register
ICV_DIR_EL1	3	0	C12	C11	1	–	64-bit	Interrupt Controller Deactivate Virtual Interrupt Register
ICC_RPR_EL1	3	0	C12	C11	3	–	64-bit	Interrupt Controller Running Priority Register
ICV_RPR_EL1	3	0	C12	C11	3	–	64-bit	Interrupt Controller Virtual Running Priority Register
ICC_SGI1R_EL1	3	0	C12	C11	5	–	64-bit	Interrupt Controller Software Generated Interrupt Group 1 Register
ICC_ASGI1R_EL1	3	0	C12	C11	6	–	64-bit	Interrupt Controller Alias Software Generated Interrupt Group 1 Register

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

13. Advanced SIMD and floating-point support

The Cortex®-X3 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping. The Cortex®-X3 core floating-point implementation includes Arm®v8.3-A and Arm®v8.5-A features.

The Cortex®-X3 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 Cortex®-X3 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 Cortex®-X3 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*.

15. System control

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

The main functions of the system registers are:

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

The system registers are accessible in AArch64 execution state at EL0 to EL3.

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

15.1 AArch64 Identification register summary

The summary table provides an overview of all Identification registers in the core. For more information about a register, click the register name in the table.

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

Table 15-1: Identification registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ID_ISAR3_EL1	3	0	C0	C2	3	–	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	0	C0	C2	4	–	64-bit	AArch32 Instruction Set Attribute Register 4
ID_ISAR5_EL1	3	0	C0	C2	5	–	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	0	C0	C2	6	–	64-bit	AArch32 Memory Model Feature Register 4
ID_ISAR6_EL1	3	0	C0	C2	7	–	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	0	C0	C3	0	–	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	0	C0	C3	1	–	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	0	C0	C3	2	–	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	0	C0	C3	4	–	64-bit	AArch32 Processor Feature Register 2
ID_DFR1_EL1	3	0	C0	C3	5	–	64-bit	Debug Feature Register 1
ID_MMFR5_EL1	3	0	C0	C3	6	–	64-bit	AArch32 Memory Model Feature Register 5
ID_AA64PFR0_EL1	3	0	C0	C4	0	–	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	0	C0	C4	1	–	64-bit	AArch64 Processor Feature Register 1
ID_AA64PFR2_EL1	3	0	C0	C4	2	–	64-bit	AArch64 Processor Feature Register 2
ID_AA64ZFR0_EL1	3	0	C0	C4	4	–	64-bit	SVE Feature ID register 0
ID_AA64DFR0_EL1	3	0	C0	C5	0	–	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	0	C0	C5	1	–	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	0	C0	C5	4	–	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	0	C0	C5	5	–	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	0	C0	C6	0	–	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	0	C0	C6	1	–	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64ISAR2_EL1	3	0	C0	C6	2	–	64-bit	AArch64 Instruction Set Attribute Register 2
ID_AA64MMFR0_EL1	3	0	C0	C7	0	–	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	0	C0	C7	1	–	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	0	C0	C7	2	–	64-bit	AArch64 Memory Model Feature Register 2
MPAMIDR_EL1	3	0	C10	C4	4	–	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	0	C15	C0	0	–	64-bit	CPU Configuration Register
CCSIDR_EL1	3	1	C0	C0	0	–	64-bit	Current Cache Size ID Register
CLIDR_EL1	3	1	C0	C0	1	–	64-bit	Cache Level ID Register
CCSIDR2_EL1	3	1	C0	C0	2	–	64-bit	Current Cache Size ID Register 2
GMID_EL1	3	1	C0	C0	4	–	64-bit	Multiple tag transfer ID register
CSSELR_EL1	3	2	C0	C0	0	–	64-bit	Cache Size Selection Register
CTR_EL0	3	3	C0	C0	1	–	64-bit	Cache Type Register
DCZID_EL0	3	3	C0	C0	7	–	64-bit	Data Cache Zero ID register
VPIDR_EL2	3	4	C0	C0	0	–	64-bit	Virtualization Processor ID Register
VMPIDR_EL2	3	4	C0	C0	5	–	64-bit	Virtualization Multiprocessor ID Register

16. Debug

The DynamIQ™-110 cluster provides a debug system that supports both self-hosted and external debug. It has an external DebugBlock component, and integrates various CoreSight debug related components.

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

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

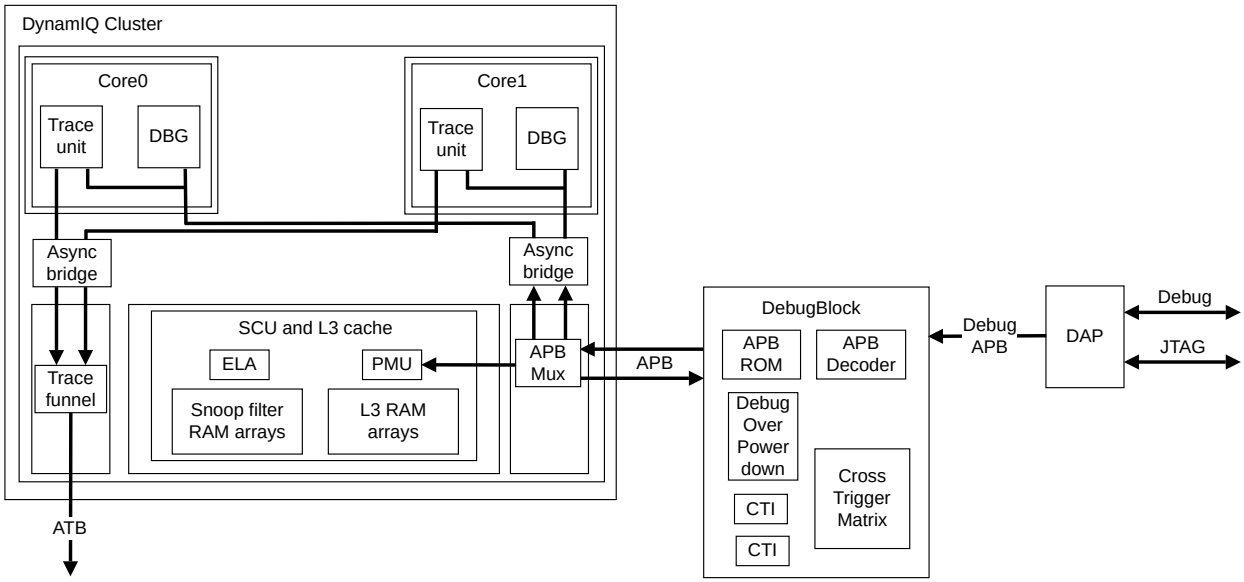
The connection between the cluster and the DebugBlock consists of a pair of *Advanced Peripheral Bus* APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register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

The debug system implements the following CoreSight debug components:

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

The following figure shows how the debug system is implemented with the DynamIQ™ cluster.

Figure 16-1: DynamIQ™ cluster debug components



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

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

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

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

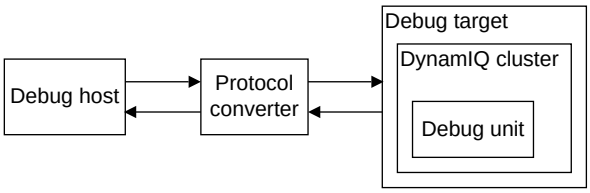
The Cortex®-X3 core also supports direct access to internal memory, that is, cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use. See 10. Direct access to internal memory on page 75 for more information.

16.1 Supported debug methods

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

The following figure shows a typical external debug system.

Figure 16-2: External debug system



Debug host

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

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit. For DSU-110 based devices, the mechanism used to access the debug unit is based on the CoreSight architecture. The DSU-110 DebugBlock is accessed using an APB interface and the debug accesses are then directed to the selected X3 core inside the DynamIQ™ cluster. An example of a debug target is a development system with a test chip or a silicon part with a X3 core.

Debug unit

Helps debugging software that is running on the core:

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

With the debug unit, you can:

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

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

16.2 Debug register interfaces

The Cortex®-X3 core implements the Arm®v9.0-A Debug architecture. It also supports the Arm®v8.4-A Debug architecture and the 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.

Related information

5.7 Debug over powerdown on page 55

16.2.1 Core interfaces

System register access allows the Cortex®-X3 core to access certain Debug registers directly. The Debug register interfaces provide access to these registers either from software running on the core or from an external debugger.

Access to the Debug registers is partitioned as follows:

Debug

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

Performance monitoring

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

Trace

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

Statistical profiling

This function is system register based.

ELA registers

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

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

16.2.2 Effects of resets on debug registers

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

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

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

16.2.3 External access permissions to Debug registers

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

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

Table 16-1: External access conditions to registers

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

Name	Condition	Description
EDAD	`AllowExternalDebugAccess()==FALSE`	External debug access is disabled. If an error is returned because of an EDAD condition code, and this is the highest priority error condition, then EDPRSR.SDAD is set to 1. If not, then SDAD is unchanged.
Default	-	This is normal access, none of the conditions apply.

16.2.4 Breakpoints and watchpoints

The Cortex®-X3 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 and the other two (BRP 4 and 5) match against either the *Virtual Address* (VA) or context ID, or the *Virtual Machine ID* (VMID). All the watchpoints can be linked to two breakpoints (BRP 4 and 5) to enable a memory request to be trapped in a given process context.

16.3 Debug events

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

The Cortex®-X3 core responds to a debug event in one of the following ways:

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

In the Cortex®-X3 core, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations, except `DC ZVA`, and `DC IVAC` do not generate watchpoint debug events. Store exclusive instructions generate a watchpoint debug event even when the check for the control of exclusive monitor fails. Atomic `CAS` instructions generate a watchpoint debug event even when the compare operation fails.

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

16.4 Debug memory map and debug signals

The debug memory map and debug signals are handled at cluster level.

See the *Debug* chapter of the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

16.5 ROM table

The Cortex®-X3 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 Cortex®-X3 core. There is one ROM table for each core and ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

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

The Cortex®-X3 core ROM table includes the following entries:

Table 16-2: Core ROM table

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

Related information

B.1 External CoreROM registers summary on page 960

16.6 CoreSight component identification

Each component associated with the Cortex®-X3 core has a unique set of CoreSight ID values.

Table 16-3: Cortex®-X3 CoreSight component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Revision
Trace unit	0x04201BBD4E	0xB105900D	0x00000013	0x47705A13	r1p2
PMU	0x04201BBD4E	0xB105900D	0x00000016	0x47702A16	r1p2
DBG	0x04201BBD4E	0xB105900D	0x00000015	0x47709A15	r1p2
ROM Table	0x04201BBD4E	0xB105900D	0x00000000	0x47700AF7	r1p2

For details on the CoreSight component identification for the Cortex®-X3 core ELA, see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

16.7 CTI register identification values

The Cortex®-X3 core Cross Trigger Interface (CTI) registers are located in the DebugBlock of the DSU-110.

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

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

Table 16-4: Core CTI register peripheral ID values

Register	Bitfield position	Bitfield name	Value
CTIPIDR4	[7:4]	SIZE	See register B.3.54 PMPIDR4, Performance Monitors Peripheral Identification Register 4 on page 1106
	[3:0]	DES_2	
CTIPIDR3	[7:4]	REVAND	See register B.3.58 PMPIDR3, Performance Monitors Peripheral Identification Register 3 on page 1112
	[3:0]	CMOD	
CTIPIDR2	[7:4]	REVISION	See register B.3.57 PMPIDR2, Performance Monitors Peripheral Identification Register 2 on page 1111
	[3]	JEDEC	
	[2:0]	DES_1	
CTIPIDR1	[7:4]	DES_0	See register B.3.56 PMPIDR1, Performance Monitors Peripheral Identification Register 1 on page 1109
	[3:0]	PART_1	
CTIPIDR0	[7:0]	PART_0	See register B.3.55 PMPIDR0, Performance Monitors Peripheral Identification Register 0 on page 1108

16.8 AArch64 Debug register summary

The summary table provides an overview of all Debug registers in the core. For more information about a register, click the register name in the table.

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

Table 16-5: Debug registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
OSDTRRX_EL1	2	0	C0	C0	2	—	64-bit	OS Lock Data Transfer Register, Receive
DBGBVR0_EL1	2	0	C0	C0	4	—	64-bit	Debug Breakpoint Value Registers
DBGBCR0_EL1	2	0	C0	C0	5	—	64-bit	Debug Breakpoint Control Registers

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
DBGWVR0_EL1	2	0	C0	C0	6	–	64-bit	Debug Watchpoint Value Registers
DBGWCR0_EL1	2	0	C0	C0	7	–	64-bit	Debug Watchpoint Control Registers
DBGBVR1_EL1	2	0	C0	C1	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR1_EL1	2	0	C0	C1	5	–	64-bit	Debug Breakpoint Control Registers
DBGWVR1_EL1	2	0	C0	C1	6	–	64-bit	Debug Watchpoint Value Registers
DBGWCR1_EL1	2	0	C0	C1	7	–	64-bit	Debug Watchpoint Control Registers
MDCCINT_EL1	2	0	C0	C2	0	–	64-bit	Monitor DCC Interrupt Enable Register
MDSCR_EL1	2	0	C0	C2	2	–	64-bit	Monitor Debug System Control Register
DBGBVR2_EL1	2	0	C0	C2	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR2_EL1	2	0	C0	C2	5	–	64-bit	Debug Breakpoint Control Registers
DBGWVR2_EL1	2	0	C0	C2	6	–	64-bit	Debug Watchpoint Value Registers
DBGWCR2_EL1	2	0	C0	C2	7	–	64-bit	Debug Watchpoint Control Registers
OSDTRTX_EL1	2	0	C0	C3	2	–	64-bit	OS Lock Data Transfer Register, Transmit
DBGBVR3_EL1	2	0	C0	C3	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR3_EL1	2	0	C0	C3	5	–	64-bit	Debug Breakpoint Control Registers
DBGWVR3_EL1	2	0	C0	C3	6	–	64-bit	Debug Watchpoint Value Registers
DBGWCR3_EL1	2	0	C0	C3	7	–	64-bit	Debug Watchpoint Control Registers
DBGBVR4_EL1	2	0	C0	C4	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR4_EL1	2	0	C0	C4	5	–	64-bit	Debug Breakpoint Control Registers
DBGBVR5_EL1	2	0	C0	C5	4	–	64-bit	Debug Breakpoint Value Registers
DBGBCR5_EL1	2	0	C0	C5	5	–	64-bit	Debug Breakpoint Control Registers
OSECCR_EL1	2	0	C0	C6	2	–	64-bit	OS Lock Exception Catch Control Register
MDRAR_EL1	2	0	C1	C0	0	–	64-bit	Monitor Debug ROM Address Register
OSLAR_EL1	2	0	C1	C0	4	–	64-bit	OS Lock Access Register
OSLSR_EL1	2	0	C1	C1	4	–	64-bit	OS Lock Status Register
OSDLR_EL1	2	0	C1	C3	4	–	64-bit	OS Double Lock Register
DBGPRCR_EL1	2	0	C1	C4	4	–	64-bit	Debug Power Control Register
DBGCLAIMSET_EL1	2	0	C7	C8	6	–	64-bit	Debug CLAIM Tag Set register
DBGCLAIMCLR_EL1	2	0	C7	C9	6	–	64-bit	Debug CLAIM Tag Clear register
DBGAUTHSTATUS_EL1	2	0	C7	C14	6	–	64-bit	Debug Authentication Status register
MDCCSR_EL0	2	3	C0	C1	0	–	64-bit	Monitor DCC Status Register
DBGDTR_EL0	2	3	C0	C4	0	–	64-bit	Debug Data Transfer Register, half-duplex
DBGDTRRX_EL0	2	3	C0	C5	0	–	64-bit	Debug Data Transfer Register, Receive
DBGDTRTX_EL0	2	3	C0	C5	0	–	64-bit	Debug Data Transfer Register, Transmit
TRFCR_EL1	3	0	C1	C2	1	–	64-bit	Trace Filter Control Register (EL1)
MDCR_EL2	3	4	C1	C1	1	–	64-bit	Monitor Debug Configuration Register (EL2)
TRFCR_EL2	3	4	C1	C2	1	–	64-bit	Trace Filter Control Register (EL2)
IMP_IDATA0_EL3	3	6	C15	C0	0	–	64-bit	Instruction Register 0
IMP_IDATA1_EL3	3	6	C15	C0	1	–	64-bit	Instruction Register 0
IMP_IDATA2_EL3	3	6	C15	C0	2	–	64-bit	Instruction Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_DDATA0_EL3	3	6	C15	C1	0	–	64-bit	Data Register 0
IMP_DDATA1_EL3	3	6	C15	C1	1	–	64-bit	Data Register 1
IMP_DDATA2_EL3	3	6	C15	C1	2	–	64-bit	Data Register 2

16.9 External Debug register summary

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

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

Table 16-6: Debug registers summary

Offset	Name	Reset	Width	Description
0x020	EDESR	–	32-bit	External Debug Event Status Register
0x024	EDECR	–	32-bit	External Debug Execution Control Register
0x030	EDWAR [31:0]	–	32-bit	External Debug Watchpoint Address Register
0x034	EDWAR [63:32]	–	32-bit	External Debug Watchpoint Address Register
0x080	DBGDTRRX_EL0	–	32-bit	Debug Data Transfer Register, Receive
0x084	EDITR	–	32-bit	External Debug Instruction Transfer Register
0x088	EDSCR	–	32-bit	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	–	32-bit	Debug Data Transfer Register, Transmit
0x090	EDRCR	–	32-bit	External Debug Reserve Control Register
0x098	EDECCR	–	32-bit	External Debug Exception Catch Control Register
0x300	OSLAR_EL1	–	32-bit	OS Lock Access Register
0x310	EDPRCR	–	32-bit	External Debug Power/Reset Control Register
0x314	EDPRSR	–	32-bit	External Debug Processor Status Register
0x400	DBGBVR0_EL1 [63:0]	–	64-bit	Debug Breakpoint Value Registers
0x408	DBGBCR0_EL1	–	32-bit	Debug Breakpoint Control Registers
0x410	DBGBVR1_EL1 [63:0]	–	64-bit	Debug Breakpoint Value Registers
0x418	DBGBCR1_EL1	–	32-bit	Debug Breakpoint Control Registers
0x420	DBGBVR2_EL1 [63:0]	–	64-bit	Debug Breakpoint Value Registers
0x428	DBGBCR2_EL1	–	32-bit	Debug Breakpoint Control Registers
0x430	DBGBVR3_EL1 [63:0]	–	64-bit	Debug Breakpoint Value Registers
0x438	DBGBCR3_EL1	–	32-bit	Debug Breakpoint Control Registers
0x440	DBGBVR4_EL1 [63:0]	–	64-bit	Debug Breakpoint Value Registers
0x448	DBGBCR4_EL1	–	32-bit	Debug Breakpoint Control Registers
0x450	DBGBVR5_EL1 [63:0]	–	64-bit	Debug Breakpoint Value Registers
0x458	DBGBCR5_EL1	–	32-bit	Debug Breakpoint Control Registers
0x800	DBGWVR0_EL1 [63:0]	–	64-bit	Debug Watchpoint Value Registers

Offset	Name	Reset	Width	Description
0x808	DBGWCR0_EL1	—	32-bit	Debug Watchpoint Control Registers
0x810	DBGWVR1_EL1 [63:0]	—	64-bit	Debug Watchpoint Value Registers
0x818	DBGWCR1_EL1	—	32-bit	Debug Watchpoint Control Registers
0x820	DBGWVR2_EL1 [63:0]	—	64-bit	Debug Watchpoint Value Registers
0x828	DBGWCR2_EL1	—	32-bit	Debug Watchpoint Control Registers
0x830	DBGWVR3_EL1 [63:0]	—	64-bit	Debug Watchpoint Value Registers
0x838	DBGWCR3_EL1	—	32-bit	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	—	32-bit	Main ID Register
0xD20	EDPFR [31:0]	—	32-bit	External Debug Processor Feature Register
0xD24	EDPFR [63:32]	—	32-bit	External Debug Processor Feature Register
0xD28	EDDFR [31:0]	—	32-bit	External Debug Feature Register
0xD2C	EDDFR [63:32]	—	32-bit	External Debug Feature Register
0xD60	EDAA32PFR	—	64-bit	External Debug Auxiliary Processor Feature Register
0xF00	EDITCTRL	—	32-bit	External Debug Integration mode Control register
0xFA0	DBGCLAIMSET_EL1	—	32-bit	Debug CLAIM Tag Set register
0xFA4	DBGCLAIMCLR_EL1	—	32-bit	Debug CLAIM Tag Clear register
0xFA8	EDDEVAFF0	—	32-bit	External Debug Device Affinity register 0
0xFAC	EDDEVAFF1	—	32-bit	External Debug Device Affinity register 1
0xFB0	EDLAR	—	32-bit	External Debug Lock Access Register
0xFB4	EDLSR	—	32-bit	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	—	32-bit	Debug Authentication Status register
0xFBC	EDDEVARCH	—	32-bit	External Debug Device Architecture register
0xFC0	EDDEVID2	—	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	—	32-bit	External Debug Device ID register 1
0xFC8	EDDEVID	—	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	—	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	—	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	—	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	—	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	—	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	—	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	—	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	—	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	—	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	—	32-bit	External Debug Component Identification Register 3

16.10 External CoreROM register summary

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

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

Table 16-7: CoreROM registers summary

Offset	Name	Reset	Width	Description
0x000	COREROM_ROMENTRY0	—	32-bit	Core ROM table Entry 0
0x004	COREROM_ROMENTRY1	—	32-bit	Core ROM table Entry 1
0x008	COREROM_ROMENTRY2	—	32-bit	Core ROM table Entry 2
0x00C	COREROM_ROMENTRY3	—	32-bit	Core ROM table Entry 3
0xFB8	COREROM_AUTHSTATUS	—	32-bit	Core ROM table Authentication Status Register
0xFBC	COREROM_DEVARCH	—	32-bit	Core ROM table Device Architecture Register
0xFCC	COREROM_DEVTYPE	—	32-bit	Core ROM table Device Type Register
0xFD0	COREROM_PIDR4	—	32-bit	Core ROM table Peripheral Identification Register 4
0xFE0	COREROM_PIDR0	—	32-bit	Core ROM table Peripheral Identification Register 0
0xFE4	COREROM_PIDR1	—	32-bit	Core ROM table Peripheral Identification Register 1
0xFE8	COREROM_PIDR2	—	32-bit	Core ROM table Peripheral Identification Register 2
0xFEC	COREROM_PIDR3	—	32-bit	Core ROM table Peripheral Identification Register 3
0xFF0	COREROM_CIDR0	—	32-bit	Core ROM table Component Identification Register 0
0xFF4	COREROM_CIDR1	—	32-bit	Core ROM table Component Identification Register 1
0xFF8	COREROM_CIDR2	—	32-bit	Core ROM table Component Identification Register 2
0xFFC	COREROM_CIDR3	—	32-bit	Core ROM table Component Identification Register 3

17. Performance Monitors Extension support

The Cortex®-X3 core implements the Performance Monitors Extension, including Arm®v8.4-A and Arm®v8.5-A performance monitoring features.

The Cortex®-X3 core *Performance Monitoring Unit* (PMU):

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

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

17.1 Performance monitors events

The Cortex®-X3 core *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

The following table lists the Cortex®-X3 core 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 trace unit and selected in accordance with the *Arm® Embedded Trace Extension*.

Table 17-1: 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).
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

Event number	Event mnemonic	Event description
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.
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.

Event number	Event mnemonic	Event description
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
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
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.

Event number	Event mnemonic	Event description
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_CYCLES	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.
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.

Event number	Event mnemonic	Event description
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.
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.

Event number	Event mnemonic	Event description
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
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

Event number	Event mnemonic	Event description
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
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.

Event number	Event mnemonic	Event description
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.
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

Event number	Event mnemonic	Event description
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
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

Event number	Event mnemonic	Event description
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
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 This event is not exported to the trace unit.
0x4011	TRCEXTOUT1	PE Trace Unit external output 1 This event is not exported to the trace unit.
0x4012	TRCEXTOUT2	PE Trace Unit external output 2 This event is not exported to the trace unit.
0x4013	TRCEXTOUT3	PE Trace Unit external output 3 This event is not exported to the trace unit.
0x4018	CTI_TRIGOUT4	Cross-trigger Interface output trigger 4
0x4019	CTI_TRIGOUT5	Cross-trigger Interface output trigger 5
0x401A	CTI_TRIGOUT6	Cross-trigger Interface output trigger 6
0x401B	CTI_TRIGOUT7	Cross-trigger Interface output trigger 7
0x4020	LDST_ALIGN_LAT	Access with additional latency from alignment
0x4021	LD_ALIGN_LAT	Load with additional latency from alignment
0x4022	ST_ALIGN_LAT	Store with additional latency from alignment
0x4024	MEM_ACCESS_CHECKED	Checked data memory access
0x4025	MEM_ACCESS_RD_CHECKED	Checked data memory access, read
0x4026	MEM_ACCESS_WR_CHECKED	Checked data memory access, write
0x8005	ASE_INST_SPEC	Advanced SIMD operations speculatively executed
0x8006	SVE_INST_SPEC	SVE 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

Event number	Event mnemonic	Event description
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

17.2 Performance monitors interrupts

Performance monitors interrupts indicate events that have been observed several times.

When the *Performance Monitoring Unit* (PMU) generates an interrupt, the nPMUIRQ[n] output is driven LOW.

See the *Performance Monitors Extension support* chapter of the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for more information.

17.3 External register access permissions

The Cortex®-X3 core supports access to the *Performance Monitoring Unit* (PMU) registers from the system register interface and a memory-mapped interface.

Access to a register depends on:

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

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

17.4 AArch64 Performance Monitors register summary

The summary table provides an overview of all Performance Monitors registers in the core. For more information about a register, click the register name in the table.

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

Table 17-2: Performance Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMINTENSET_EL1	3	0	C9	C14	1	–	64-bit	Performance Monitors Interrupt Enable Set register
PMINTENCLR_EL1	3	0	C9	C14	2	–	64-bit	Performance Monitors Interrupt Enable Clear register
PMMIR_EL1	3	0	C9	C14	6	–	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	3	C9	C12	0	–	64-bit	Performance Monitors Control Register
PMCNTENSET_EL0	3	3	C9	C12	1	–	64-bit	Performance Monitors Count Enable Set register
PMCNTENCLR_EL0	3	3	C9	C12	2	–	64-bit	Performance Monitors Count Enable Clear register
PMOVSCLR_EL0	3	3	C9	C12	3	–	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_EL0	3	3	C9	C12	4	–	64-bit	Performance Monitors Software Increment register
PMSELR_EL0	3	3	C9	C12	5	–	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	3	3	C9	C12	6	–	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	3	C9	C12	7	–	64-bit	Performance Monitors Common Event Identification register 1
PMCCNTR_EL0	3	3	C9	C13	0	–	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	–	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	–	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	3	3	C9	C14	0	–	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	–	64-bit	Performance Monitors Overflow Flag Status Set register
PMEVCNTR0_EL0	3	3	C14	C8	0	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	–	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	–	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	–	64-bit	Performance Monitors Cycle Count Filter Register

17.5 External PMU register summary

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

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

Table 17-3: PMU registers summary

Offset	Name	Reset	Width	Description
0x0	PMEVCNTR0_EL0	—	64-bit	Performance Monitors Event Count Registers
0x8	PMEVCNTR1_EL0	—	64-bit	Performance Monitors Event Count Registers
0x10	PMEVCNTR2_EL0	—	64-bit	Performance Monitors Event Count Registers
0x18	PMEVCNTR3_EL0	—	64-bit	Performance Monitors Event Count Registers
0x20	PMEVCNTR4_EL0	—	64-bit	Performance Monitors Event Count Registers
0x28	PMEVCNTR5_EL0	—	64-bit	Performance Monitors Event Count Registers
0x40	PMEVCNTR8_EL0	—	64-bit	Performance Monitors Event Count Registers
0x60	PMEVCNTR12_EL0	—	64-bit	Performance Monitors Event Count Registers
0x80	PMEVCNTR16_EL0	—	64-bit	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_EL0 [31:0]	—	32-bit	Performance Monitors Cycle Counter
0x0FC	PMCCNTR_EL0 [63:32]	—	32-bit	Performance Monitors Cycle Counter
0x200	PMPCSR [31:0]	—	32-bit	Program Counter Sample Register
0x204	PMPCSR [63:32]	—	32-bit	Program Counter Sample Register
0x220	PMPCSR [31:0]	—	32-bit	Program Counter Sample Register
0x224	PMPCSR [63:32]	—	32-bit	Program Counter Sample Register
0x208	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x228	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	—	32-bit	VMID Sample Register
0x22C	PMCID2SR	—	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0	—	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	—	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	—	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	—	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register

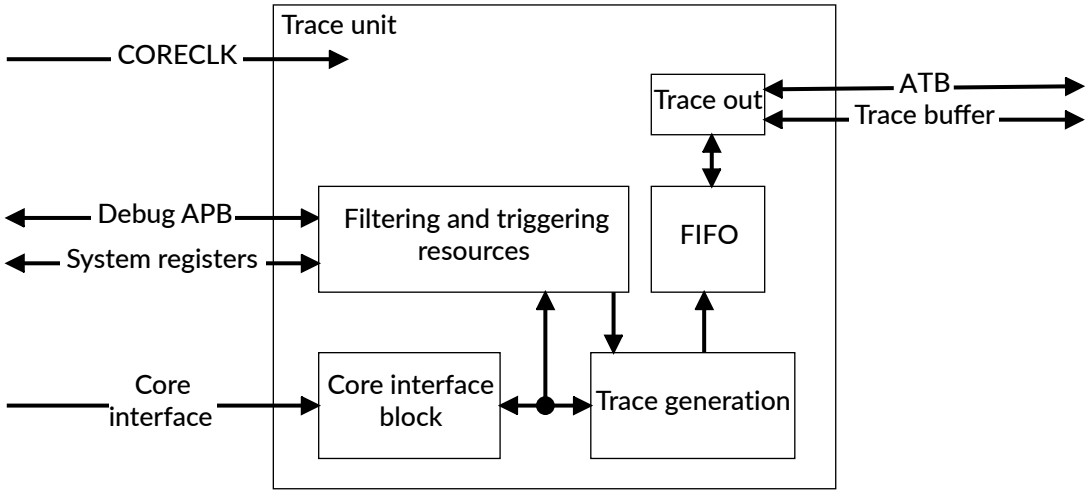
Offset	Name	Reset	Width	Description
0x610	PMSSSR	—	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	—	64-bit	PMU Cycle Counter Snapshot Register
0x620 + (8 * n)	PMEVCNTSR_n_	—	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMSSCR	—	32-bit	PMU Snapshot Capture Register
0xC00	PMCNTENSET_EL0	—	32-bit	Performance Monitors Count Enable Set register
0xC20	PMCNTENCLR_EL0	—	32-bit	Performance Monitors Count Enable Clear register
0xC40	PMINTENSET_EL1	—	32-bit	Performance Monitors Interrupt Enable Set register
0xC60	PMINTENCLR_EL1	—	32-bit	Performance Monitors Interrupt Enable Clear register
0xC80	PMOVSCLR_EL0	—	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCA0	PMSWINC_EL0	—	32-bit	Performance Monitors Software Increment register
0xCC0	PMOVSSET_EL0	—	32-bit	Performance Monitors Overflow Flag Status Set register
0xE00	PMCFGR	—	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	—	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	—	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	—	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	—	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	—	32-bit	Performance Monitors Common Event Identification register 3
0xE40	PMMIR	—	32-bit	Performance Monitors Machine Identification Register
0xFA8	PMDEVAFF0	—	32-bit	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	—	32-bit	Performance Monitors Device Affinity register 1
0xFB0	PMLAR	—	32-bit	Performance Monitors Lock Access Register
0xFB4	PMLSR	—	32-bit	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	—	32-bit	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	—	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	—	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	—	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	—	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	—	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	—	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	—	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	—	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	—	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	—	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	—	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	—	32-bit	Performance Monitors Component Identification Register 3

18. Embedded Trace Extension support

The Cortex®-X3 core implements the *Embedded Trace Extension* (ETE). The trace unit performs real-time instruction flow tracing based on the ETE. The trace unit is a CoreSight component and is an integral part of the Arm Real-time Debug solution, the Arm Debugger. The Arm Debugger is a part of the Arm Development Studio.

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

Figure 18-1: 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 trace unit generates by filtering. For example, you can limit trace generation to a certain address range. The trace unit supports other, more complicated, logic analyzer style filtering options. The trace unit can also generate a trigger that is a signal to the Trace Capture Device to stop capturing trace.

FIFO

The trace unit generates trace in a highly compressed form. The FIFO enables trace bursts to be flattened out. When the FIFO is full, the FIFO signals an overflow. The trace generation logic does not generate any new trace until the FIFO is emptied. This behavior causes a gap in the trace when viewed in the debugger.

Trace out

Trace from the FIFO is output on the AMBA ATB interface or to the trace buffer.

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

18.1 Trace unit resources

Trace resources include counters, external inputs and outputs, and comparators.

The following table shows the trace unit resources, and indicates which of these resources Cortex®-X3 core implements.

Table 18-1: Trace unit resources implemented

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

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

18.2 Trace unit generation options

The Cortex®-X3 core trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the Cortex®-X3 core trace unit.

Table 18-2: Trace unit generation options implemented

Description	Configuration
Instruction address size in bytes	8

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

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

18.3 Reset the trace unit

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

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

18.4 Program and read the trace unit registers

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

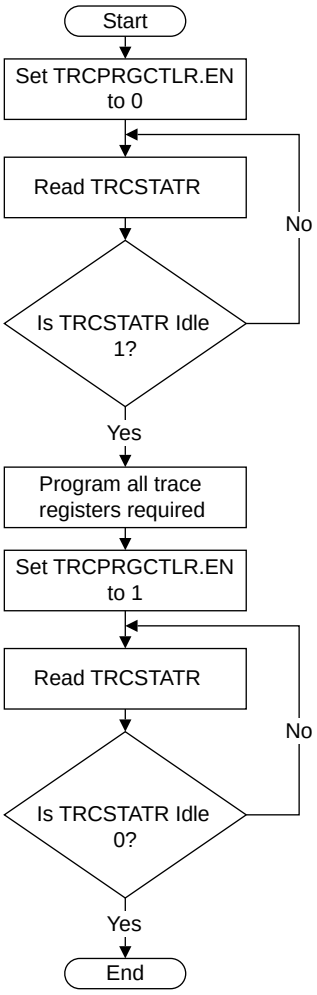
The core does not have to be in debug state when you program the trace unit registers. When you program the trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place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

For more information on the following registers, see the *Arm® Architecture Reference Manual for A-profile architecture*:

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

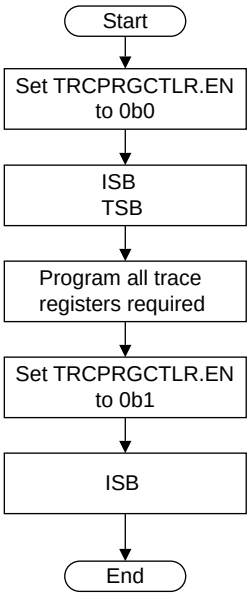
The following figure shows the flow for programming trace unit registers using the Debug APB interface:

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



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

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



18.5 Trace unit register interfaces

The Cortex®-X3 core supports an APB memory-mapped interface and a system register interface to trace unit registers.

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

18.6 Interaction with the Performance Monitoring Unit and Debug

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

Interaction with the PMU

The Cortex®-X3 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and trace unit function together.

Use of PMU events by the trace unit

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

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

Related information

17. Performance Monitors Extension support on page 115
17.1 Performance monitors events on page 115

18.7 ETE events

The Cortex®-X3 core trace unit collects events from other units in the design and uses numbers to reference these events.

Other than the events mentioned in 17.1 Performance monitors events on page 115, the following events are also exported:

Table 18-3: 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_HOVFS	PMU overflow, counters reserved for use by EL2

18.8 AArch64 Trace unit register summary

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

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

Table 18-4: Trace unit registers summary

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

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

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

18.9 External ETE register summary

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

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

Table 18-5: ETE registers summary

Offset	Name	Reset	Width	Description
0x004	TRCPRGCTLR	—	32-bit	Programming Control Register
0x00C	TRCSTATR	—	32-bit	Trace Status Register
0x010	TRCCONFIGR	—	32-bit	Trace Configuration Register
0x018	TRCAUXCTLR	—	32-bit	Auxiliary Control Register
0x020	TRCEVENTCTL0R	—	32-bit	Event Control 0 Register
0x024	TRCEVENTCTL1R	—	32-bit	Event Control 1 Register
0x028	TRCRSR	—	32-bit	Resources Status Register
0x030	TRCTSCTLR	—	32-bit	Timestamp Control Register
0x034	TRCSYNCPR	—	32-bit	Synchronization Period Register
0x038	TRCCCCTLR	—	32-bit	Cycle Count Control Register
0x03C	TRCBBCTLR	—	32-bit	Branch Broadcast Control Register
0x040	TRCTRACEIDR	—	32-bit	Trace ID Register
0x080	TRCVICTLR	—	32-bit	ViewInst Main Control Register
0x084	TRCVIIECTLR	—	32-bit	ViewInst Include/Exclude Control Register
0x088	TRCVISSCTLR	—	32-bit	ViewInst Start/Stop Control Register
0x100	TRCSEQEVR0	—	32-bit	Sequencer State Transition Control Register <n>
0x104	TRCSEQEVR1	—	32-bit	Sequencer State Transition Control Register <n>
0x108	TRCSEQEVR2	—	32-bit	Sequencer State Transition Control Register <n>
0x118	TRCSEQRSTEVR	—	32-bit	Sequencer Reset Control Register
0x11C	TRCSEQSTR	—	32-bit	Sequencer State Register
0x120	TRCEXTINSELR0	—	32-bit	External Input Select Register <n>
0x124	TRCEXTINSELR1	—	32-bit	External Input Select Register <n>
0x128	TRCEXTINSELR2	—	32-bit	External Input Select Register <n>
0x12C	TRCEXTINSELR3	—	32-bit	External Input Select Register <n>
0x140	TRCCNTRLDVR0	—	32-bit	Counter Reload Value Register <n>
0x144	TRCCNTRLDVR1	—	32-bit	Counter Reload Value Register <n>
0x150	TRCCNTCTLR0	—	32-bit	Counter Control Register <n>
0x154	TRCCNTCTLR1	—	32-bit	Counter Control Register <n>
0x160	TRCCNTVR0	—	32-bit	Counter Value Register <n>
0x164	TRCCNTVR1	—	32-bit	Counter Value Register <n>
0x180	TRCIDR8	—	32-bit	ID Register 8
0x184	TRCIDR9	—	32-bit	ID Register 9

Offset	Name	Reset	Width	Description
0x188	TRCIDR10	—	32-bit	ID Register 10
0x18C	TRCIDR11	—	32-bit	ID Register 11
0x190	TRCIDR12	—	32-bit	ID Register 12
0x194	TRCIDR13	—	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	—	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	—	32-bit	ID Register 0
0x1E4	TRCIDR1	—	32-bit	ID Register 1
0x1E8	TRCIDR2	—	32-bit	ID Register 2
0x1EC	TRCIDR3	—	32-bit	ID Register 3
0x1F0	TRCIDR4	—	32-bit	ID Register 4
0x1F4	TRCIDR5	—	32-bit	ID Register 5
0x1F8	TRCIDR6	—	32-bit	ID Register 6
0x1FC	TRCIDR7	—	32-bit	ID Register 7
0x2A0 + (4 * n)	TRCSSCSR_n_	—	32-bit	Single-shot Comparator Control Status Register <n>
0x304	TRCOSLSR	—	32-bit	Trace OS Lock Status Register
0x310	TRCPDCR	—	32-bit	PowerDown Control Register
0x314	TRCPDSR	—	32-bit	PowerDown Status Register
0x680	TRCCIDCCTLR0	—	32-bit	Context Identifier Comparator Control Register 0
0x688	TRCVMIDCCTLR0	—	32-bit	Virtual Context Identifier Comparator Control Register 0
0xF00	TRCITCTRL	—	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	—	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	—	32-bit	Claim Tag Clear Register
0xFA8	TRCDEVAFF	—	64-bit	Device Affinity Register
0xFB0	TRCLAR	—	32-bit	Lock Access Register
0xFB4	TRCLSR	—	32-bit	Lock Status Register
0xFB8	TRCAUTHSTATUS	—	32-bit	Authentication Status Register
0xFBC	TRCDEVARCH	—	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	—	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	—	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	—	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	—	32-bit	Device Type Register
0xFD0	TRCPIDR4	—	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	—	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	—	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	—	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	—	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	—	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	—	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	—	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	—	32-bit	Component Identification Register 0

Offset	Name	Reset	Width	Description
0xFF4	TRCCIDR1	—	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	—	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	—	32-bit	Component Identification Register 3

19. Trace Buffer Extension support

The Cortex®-X3 core implements the *TRace Buffer Extension* (TRBE). The TRBE writes the program flow trace generated by the trace unit directly to memory. The TRBE is programmed through System registers.

When enabled, the TRBE can:

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

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

19.1 Program and read the trace buffer registers

You 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.

19.2 Trace buffer register interface

The Cortex®-X3 core supports a system register interface to *TRace Buffer Extension* (TRBE) registers.

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

19.3 AArch64 Trace Buffer Extension register summary

The summary table provides an overview of all Trace Buffer Extension registers in the core. For more information about a register, click the register name in the table.

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

Table 19-1: Trace Buffer Extension registers summary

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

20. Activity Monitors Extension support

The Cortex®-X3 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitoring has features similar to performance monitoring features, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

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

The Cortex®-X3 core implements seven counters in two groups, each of which is a 64-bit counter that counts a fixed event. Group 0 has four counters 0-3, and Group 1 has three counters 10-12.

20.1 Activity monitors access

The Cortex®-X3 core supports access to activity monitors from the system register interface and supports read-only memory-mapped access using the utility bus interface.

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

Access enable bit

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

The CPTR_EL2.TAM bit controls access from EL0 and EL1 to the activity monitors system registers. The CPTR_EL3.TAM bit controls access from EL0, EL1, and EL2 to the Activity Monitors Extension system registers. The AMUSERENR_EL0.EN bit is configurable at EL1, EL2, and EL3. All other controls, as well as the value of the counters, are configurable only at the highest implemented Exception level. For a detailed description of access controls for the registers, see the *Arm® Architecture Reference Manual for A-profile architecture*.

System register access

The activity monitors 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 *Activity Monitoring Unit* (AMU) registers on the utility bus interface is 0x<n>90000 where “n” is the Cortex®-X3 core instance number in the DynamIQ™-110 cluster.

These registers are treated as RAZ/WI if either:

- The register is marked as Reserved.

- The register is accessed in the wrong Security state.

20.2 Activity monitors counters

The Cortex®-X3 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.

20.3 Activity monitors events

Activity monitors events in the Cortex®-X3 core are either fixed or programmable, and they map to the activity monitors counters.

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

Table 20-1: Mapping of counters to fixed events

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR00	CPU_CYCLES	`0x0011`	Core frequency cycles
AMEVCNTR01	CNT_CYCLES	`0x4004`	Constant frequency cycles
AMEVCNTR02	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	MPMM_THRESHOLD_GEAR0	`0x0300`	*Maximum Power Mitigation System* (MPMM) Gear 0 activity period threshold exceeded
AMEVCNTR11	MPMM_THRESHOLD_GEAR1	`0x0301`	*Maximum Power Mitigation System* (MPMM) Gear 1 activity period threshold exceeded

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR12	MPMM_THRESHOLD_GEAR2	0x0302	*Maximum Power Mitigation System* (MPMM) Gear 2 activity period threshold exceeded

20.4 AArch64 Activity Monitors register summary

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

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

Table 20-2: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	—	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	—	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	—	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	—	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	—	64-bit	Activity Monitors Count Enable Clear Register 0
AMCNTENSET0_EL0	3	3	C13	C2	5	—	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	—	64-bit	Activity Monitors Count Enable Clear Register 1
AMCNTENSET1_EL0	3	3	C13	C3	1	—	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR03_EL0	3	3	C13	C4	3	—	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	—	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	—	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	—	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	—	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	—	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	—	64-bit	Activity Monitors Event Type Registers 1

20.5 External AMU register summary

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

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

Table 20-3: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00 [31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0x4	AMEVCNTR00 [63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x8	AMEVCNTR01 [31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0xC	AMEVCNTR01 [63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02 [31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0x14	AMEVCNTR02 [63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03 [31:0]	—	32-bit	Activity Monitors Event Counter Registers 0
0x1C	AMEVCNTR03 [63:32]	—	32-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10 [31:0]	—	32-bit	Activity Monitors Event Counter Registers 1
0x104	AMEVCNTR10 [63:32]	—	32-bit	Activity Monitors Event Counter Registers 1
0x108	AMEVCNTR11 [31:0]	—	32-bit	Activity Monitors Event Counter Registers 1
0x10C	AMEVCNTR11 [63:32]	—	32-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12 [31:0]	—	32-bit	Activity Monitors Event Counter Registers 1
0x114	AMEVCNTR12 [63:32]	—	32-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	—	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	—	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	—	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	—	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	—	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	—	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	—	32-bit	Activity Monitors Event Type Registers 1
0x48C	AMEVTYPER13	—	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	—	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	—	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	—	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	—	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	—	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	—	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	—	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	—	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	—	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	—	32-bit	Activity Monitors Device Affinity Register 1

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

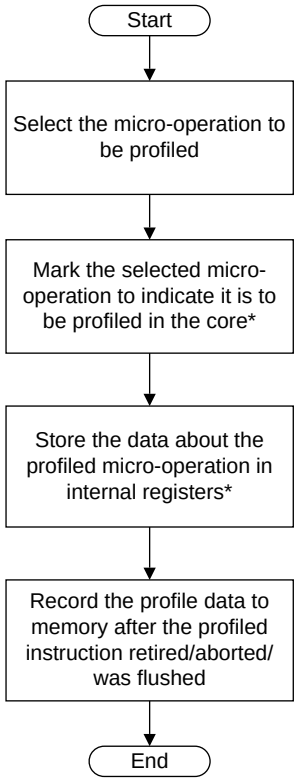
21. Statistical Profiling Extension support

The Cortex®-X3 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 Cortex®-X3 core profiles micro-operations to minimize the amount of logic necessary to support the SPE.

The following figure shows the SPE behavior in the Cortex®-X3 core.

Figure 21-1: SPE behavior



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

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

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


Note

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

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

21.1 Statistical Profiling Extension events packet

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

The following table shows the events defined in the 32-bit events packet implemented in the Cortex®-X3 core.

Table 21-1: 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

21.2 Statistical Profiling Extension data source packet

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

The following table shows the data source defined in the 8-bit data source packet implemented in the Cortex®-X3 core.

Table 21-2: SPE data source packet

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

21.3 AArch64 Statistical Profiling Extension register summary

The summary table provides an overview of all Statistical Profiling Extension registers in the core. For more information about a register, click the register name in the table.

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

Table 21-3: Statistical Profiling Extension registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSCR_EL1	3	0	C9	C9	0	–	64-bit	Statistical Profiling Control Register (EL1)
PMSICR_EL1	3	0	C9	C9	2	–	64-bit	Sampling Interval Counter Register
PMSIRR_EL1	3	0	C9	C9	3	–	64-bit	Sampling Interval Reload Register
PMSFCR_EL1	3	0	C9	C9	4	–	64-bit	Sampling Filter Control Register
PMSEVFR_EL1	3	0	C9	C9	5	–	64-bit	Sampling Event Filter Register
PMSLATFR_EL1	3	0	C9	C9	6	–	64-bit	Sampling Latency Filter Register
PMSIDR_EL1	3	0	C9	C9	7	–	64-bit	Sampling Profiling ID Register
PMBLIMITR_EL1	3	0	C9	C10	0	–	64-bit	Profiling Buffer Limit Address Register
PMBPTR_EL1	3	0	C9	C10	1	–	64-bit	Profiling Buffer Write Pointer Register
PMBSR_EL1	3	0	C9	C10	3	–	64-bit	Profiling Buffer Status/syndrome Register
PMBIDR_EL1	3	0	C9	C10	7	–	64-bit	Profiling Buffer ID Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSCR_EL2	3	4	C9	C9	0	—	64-bit	Statistical Profiling Control Register (EL2)

Appendix A AArch64 registers

This appendix contains the descriptions for the Cortex®-X3 AArch64 registers.

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

A.1 AArch64 Generic System Control registers summary

The summary table provides an overview of all Generic System Control registers in the core. For more information about a register, click the register name in the table.

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

Table A-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ACTLR_EL1	3	0	C1	C0	1	–	64-bit	Auxiliary Control Register (EL1)
RGSR_EL1	3	0	C1	C0	5	–	64-bit	Random Allocation Tag Seed Register.
GCR_EL1	3	0	C1	C0	6	–	64-bit	Tag Control Register.
TTBR0_EL1	3	0	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL1)
TTBR1_EL1	3	0	C2	C0	1	–	64-bit	Translation Table Base Register 1 (EL1)
TCR_EL1	3	0	C2	C0	2	–	64-bit	Translation Control Register (EL1)
APIAKeyLo_EL1	3	0	C2	C1	0	–	64-bit	Pointer Authentication Key A for Instruction (bits[63:0])
APIAKeyHi_EL1	3	0	C2	C1	1	–	64-bit	Pointer Authentication Key A for Instruction (bits[127:64])
APIBKeyLo_EL1	3	0	C2	C1	2	–	64-bit	Pointer Authentication Key B for Instruction (bits[63:0])
APIBKeyHi_EL1	3	0	C2	C1	3	–	64-bit	Pointer Authentication Key B for Instruction (bits[127:64])
APDAKeyLo_EL1	3	0	C2	C2	0	–	64-bit	Pointer Authentication Key A for Data (bits[63:0])
APDAKeyHi_EL1	3	0	C2	C2	1	–	64-bit	Pointer Authentication Key A for Data (bits[127:64])
APDBKeyLo_EL1	3	0	C2	C2	2	–	64-bit	Pointer Authentication Key B for Data (bits[63:0])
APDBKeyHi_EL1	3	0	C2	C2	3	–	64-bit	Pointer Authentication Key B for Data (bits[127:64])
APGAKeyLo_EL1	3	0	C2	C3	0	–	64-bit	Pointer Authentication Key A for Code (bits[63:0])
APGAKeyHi_EL1	3	0	C2	C3	1	–	64-bit	Pointer Authentication Key A for Code (bits[127:64])
SPSel	3	0	C4	C2	0	–	64-bit	Stack Pointer Select
CurrentEL	3	0	C4	C2	2	–	64-bit	Current Exception Level
PAN	3	0	C4	C2	3	–	64-bit	Privileged Access Never
UAO	3	0	C4	C2	4	–	64-bit	User Access Override
AFSR0_EL1	3	0	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	0	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL1)
ESR_EL1	3	0	C5	C2	0	–	64-bit	Exception Syndrome Register (EL1)
TFSR_EL1	3	0	C5	C6	0	–	64-bit	Tag Fault Status Register (EL1)
TFSRE0_EL1	3	0	C5	C6	1	–	64-bit	Tag Fault Status Register (EL0).

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
FAR_EL1	3	0	C6	C0	0	–	64-bit	Fault Address Register (EL1)
PAR_EL1	3	0	C7	C4	0	–	64-bit	Physical Address Register
MAIR_EL1	3	0	C10	C2	0	–	64-bit	Memory Attribute Indirection Register (EL1)
AMAIR_EL1	3	0	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
LORSA_EL1	3	0	C10	C4	0	–	64-bit	LORegion Start Address (EL1)
LOREA_EL1	3	0	C10	C4	1	–	64-bit	LORegion End Address (EL1)
LORN_EL1	3	0	C10	C4	2	–	64-bit	LORegion Number (EL1)
LORC_EL1	3	0	C10	C4	3	–	64-bit	LORegion Control (EL1)
LORID_EL1	3	0	C10	C4	7	–	64-bit	LORegionID (EL1)
VBAR_EL1	3	0	C12	C0	0	–	64-bit	Vector Base Address Register (EL1)
ISR_EL1	3	0	C12	C1	0	–	64-bit	Interrupt Status Register
CONTEXTIDR_EL1	3	0	C13	C0	1	–	64-bit	Context ID Register (EL1)
TPIDR_EL1	3	0	C13	C0	4	–	64-bit	EL1 Software Thread ID Register
SCXTNUM_EL1	3	0	C13	C0	7	–	64-bit	EL1 Read/Write Software Context Number
IMP_CPUACTLR_EL1	3	0	C15	C1	0	–	64-bit	CPU Auxiliary Control Register (EL1)
IMP_CPUACTLR2_EL1	3	0	C15	C1	1	–	64-bit	CPU Auxiliary Control Register 2 (EL1)
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	–	64-bit	CPU Auxiliary Control Register 3 (EL1)
IMP_CPUACTLR4_EL1	3	0	C15	C1	3	–	64-bit	CPU Auxiliary Control Register 4 (EL1)
IMP_CPUECTLR_EL1	3	0	C15	C1	4	–	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	0	C15	C1	5	–	64-bit	CPU Extended Control Register 2
IMP_CPUL2DIRTYLNCT_EL1	3	0	C15	C2	5	–	64-bit	CPU L2 Dirty Line Count Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	–	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	0	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register (EL1)
IMP_CPUACTLR5_EL1	3	0	C15	C8	0	–	64-bit	CPU Auxiliary Control Register 5 (EL1)
IMP_CPUACTLR6_EL1	3	0	C15	C8	1	–	64-bit	CPU Auxiliary Control Register 6 (EL1)
IMP_CPUACTLR7_EL1	3	0	C15	C8	2	–	64-bit	CPU Auxiliary Control Register 7 (EL1)
IMP_CPUACTLR8_EL1	3	0	C15	C8	5	–	64-bit	CPU Auxiliary Control Register 8 (EL1)
IMP_CPUACTLR9_EL1	3	0	C15	C8	6	–	64-bit	CPU Auxiliary Control Register 9 (EL1)
AIDR_EL1	3	1	C0	C0	7	–	64-bit	Auxiliary ID Register
NZCV	3	3	C4	C2	0	–	64-bit	Condition Flags
DAIF	3	3	C4	C2	1	–	64-bit	Interrupt Mask Bits
DIT	3	3	C4	C2	5	–	64-bit	Data Independent Timing
SSBS	3	3	C4	C2	6	–	64-bit	Speculative Store Bypass Safe
TCO	3	3	C4	C2	7	–	64-bit	Tag Check Override
FPCR	3	3	C4	C4	0	–	64-bit	Floating-point Control Register
FPSR	3	3	C4	C4	1	–	64-bit	Floating-point Status Register
TPIDR_EL0	3	3	C13	C0	2	–	64-bit	EL0 Read/Write Software Thread ID Register
TPIDRRO_EL0	3	3	C13	C0	3	–	64-bit	EL0 Read-Only Software Thread ID Register
SCXTNUM_EL0	3	3	C13	C0	7	–	64-bit	EL0 Read/Write Software Context Number
ACTLR_EL2	3	4	C1	C0	1	–	64-bit	Auxiliary Control Register (EL2)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
HACR_EL2	3	4	C1	C1	7	–	64-bit	Hypervisor Auxiliary Control Register
TTBR0_EL2	3	4	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL2)
TTBR1_EL2	3	4	C2	C0	1	–	64-bit	Translation Table Base Register 1 (EL2)
TCR_EL2	3	4	C2	C0	2	–	64-bit	Translation Control Register (EL2)
VTTBR_EL2	3	4	C2	C1	0	–	64-bit	Virtualization Translation Table Base Register
VTCR_EL2	3	4	C2	C1	2	–	64-bit	Virtualization Translation Control Register
VSTTBR_EL2	3	4	C2	C6	0	–	64-bit	Virtualization Secure Translation Table Base Register
VSTCR_EL2	3	4	C2	C6	2	–	64-bit	Virtualization Secure Translation Control Register
AFSR0_EL2	3	4	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	4	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL2)
ESR_EL2	3	4	C5	C2	0	–	64-bit	Exception Syndrome Register (EL2)
TFSR_EL2	3	4	C5	C6	0	–	64-bit	Tag Fault Status Register (EL2)
FAR_EL2	3	4	C6	C0	0	–	64-bit	Fault Address Register (EL2)
HPFAR_EL2	3	4	C6	C0	4	–	64-bit	Hypervisor IPA Fault Address Register
MAIR_EL2	3	4	C10	C2	0	–	64-bit	Memory Attribute Indirection Register (EL2)
AMAIR_EL2	3	4	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
VBAR_EL2	3	4	C12	C0	0	–	64-bit	Vector Base Address Register (EL2)
CONTEXTIDR_EL2	3	4	C13	C0	1	–	64-bit	Context ID Register (EL2)
TPIDR_EL2	3	4	C13	C0	2	–	64-bit	EL2 Software Thread ID Register
SCXTNUM_EL2	3	4	C13	C0	7	–	64-bit	EL2 Read/Write Software Context Number
IMP_ATCR_EL2	3	4	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_AVTCR_EL2	3	4	C15	C7	1	–	64-bit	CPU Virtualization Auxiliary Translation Control Register (EL2)
ACTLR_EL3	3	6	C1	C0	1	–	64-bit	Auxiliary Control Register (EL3)
SCR_EL3	3	6	C1	C1	0	–	64-bit	Secure Configuration Register
CPTR_EL3	3	6	C1	C1	2	–	64-bit	Architectural Feature Trap Register (EL3)
MDCR_EL3	3	6	C1	C3	1	–	64-bit	Monitor Debug Configuration Register (EL3)
TTBR0_EL3	3	6	C2	C0	0	–	64-bit	Translation Table Base Register 0 (EL3)
TCR_EL3	3	6	C2	C0	2	–	64-bit	Translation Control Register (EL3)
AFSR0_EL3	3	6	C5	C1	0	–	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	6	C5	C1	1	–	64-bit	Auxiliary Fault Status Register 1 (EL3)
ESR_EL3	3	6	C5	C2	0	–	64-bit	Exception Syndrome Register (EL3)
TFSR_EL3	3	6	C5	C6	0	–	64-bit	Tag Fault Status Register (EL3)
FAR_EL3	3	6	C6	C0	0	–	64-bit	Fault Address Register (EL3)
MAIR_EL3	3	6	C10	C2	0	–	64-bit	Memory Attribute Indirection Register (EL3)
AMAIR_EL3	3	6	C10	C3	0	–	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
VBAR_EL3	3	6	C12	C0	0	–	64-bit	Vector Base Address Register (EL3)
RVBAR_EL3	3	6	C12	C0	1	–	64-bit	Reset Vector Base Address Register (if EL3 implemented)
RMR_EL3	3	6	C12	C0	2	–	64-bit	Reset Management Register (EL3)
TPIDR_EL3	3	6	C13	C0	2	–	64-bit	EL3 Software Thread ID Register
SCXTNUM_EL3	3	6	C13	C0	7	–	64-bit	EL3 Read/Write Software Context Number

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPUL2SDIRTYLNCT_EL3	3	6	C15	C2	3	–	64-bit	CPU L2 Secure Dirty Line Count Register
IMP_CPUACTLR_EL3	3	6	C15	C4	0	–	64-bit	CPU Auxiliary Control Register (EL3)
IMP_ATCR_EL3	3	6	C15	C7	0	–	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_CPUPSELR_EL3	3	6	C15	C8	0	–	64-bit	Selected Instruction Private Select Register
IMP_CPUPCR_EL3	3	6	C15	C8	1	–	64-bit	Selected Instruction Private Control Register
IMP_CPUPOR_EL3	3	6	C15	C8	2	–	64-bit	Selected Instruction Private Opcode Register
IMP_CPUPMR_EL3	3	6	C15	C8	3	–	64-bit	Selected Instruction Private Mask Register
IMP_CPUPOR2_EL3	3	6	C15	C8	4	–	64-bit	Selected Instruction Private Opcode Register 2
IMP_CPUPMR2_EL3	3	6	C15	C8	5	–	64-bit	Selected Instruction Private Mask Register 2
IMP_CPUPFR_EL3	3	6	C15	C8	6	–	64-bit	Selected Instruction Private Flag Register

A.1.1 ACTLR_EL1, Auxiliary Control Register (EL1)

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


Note

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-1: AArch64_actlr_el1 bit assignments

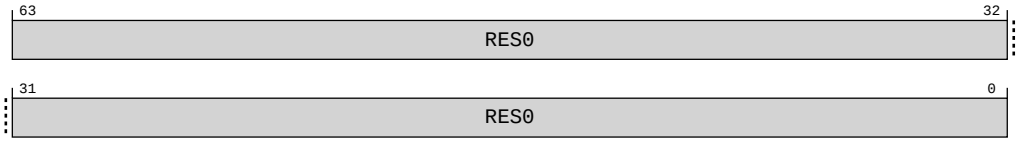


Table A-2: ACTLR_EL1 bit descriptions

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

Access

MRS <Xt>, ACTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b001

MSR ACTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b001

Accessibility

MRS <Xt>, ACTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ACTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ACTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL1;
```

MSR ACTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
```

```
    ACTLR_EL1 = X[t, 64];
```

A.1.2 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-2: AArch64_afsr0_el1 bit assignments

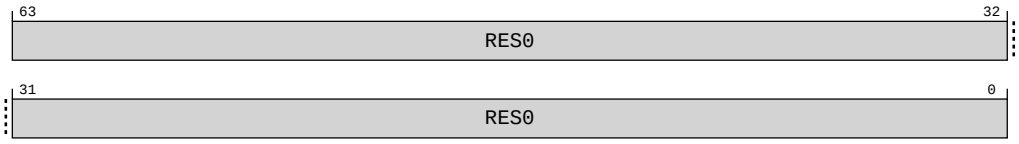


Table A-5: AFSR0_EL1 bit descriptions

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

Access

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

MRS <Xt>, AFSR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

MSR AFSR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

MRS <Xt>, AFSR0_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b000

MSR AFSR0_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b000

Accessibility

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

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL2;
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t, 64];
    else
        AFSR0_EL1 = X[t, 64];
```

```
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t, 64];
```

MRS <Xt>, AFSR0_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL1;
    else
        UNDEFINED;
```

MSR AFSR0_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t, 64];
    else
        UNDEFINED;
```

A.1.3 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-3: AArch64_afsr1_el1 bit assignments

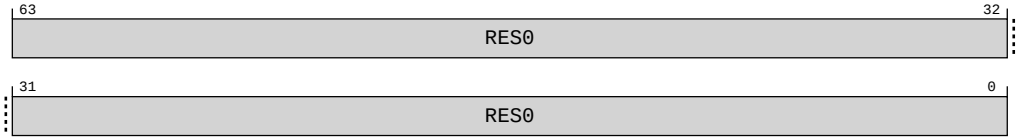


Table A-10: AFSR1_EL1 bit descriptions

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

Access

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

MRS <Xt>, AFSR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

MSR AFSR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

MRS <Xt>, AFSR1_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b001

MSR AFSR1_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0001	0b001

Accessibility

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

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL2;
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL1;
```

MSR AFSR1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t, 64];
```

MRS <Xt>, AFSR1_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL1;
    else
        UNDEFINED;
```

MSR AFSR1_EL12, <Xt>

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

```
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t, 64];
    else
        UNDEFINED;
```

A.1.4 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

AMAIR_EL1 is permitted to be cached in a TLB.

Figure A-4: AArch64_amair_el1 bit assignments

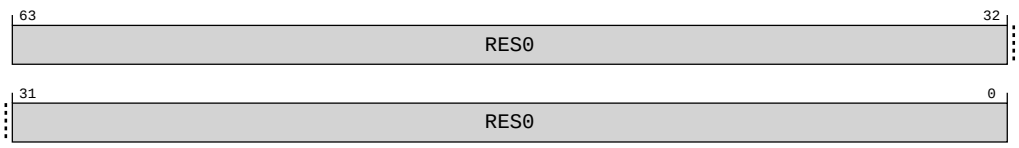


Table A-15: AMAIR_EL1 bit descriptions

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

Access

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

MRS <Xt>, AMAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

MRS <Xt>, AMAIR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b000

MSR AMAIR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0011	0b000

Accessibility

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

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR_EL2;
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL2 = X[t, 64];
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t, 64];
```

MRS <Xt>, AMAIR_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR_EL1;
    else
        UNDEFINED;
```

MSR AMAIR_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
```

A.1.5 LORID_EL1, LORegionID (EL1)

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

Configurations

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

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-5: AArch64_lorid_el1 bit assignments

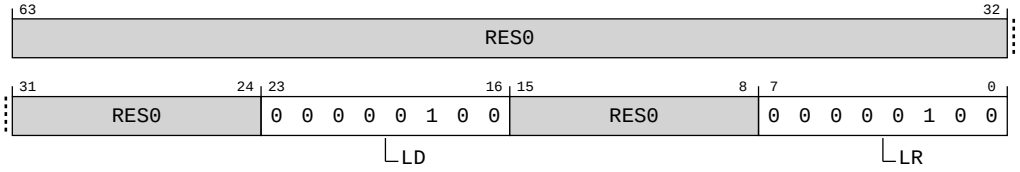


Table A-20: LORID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**
[23:16]	LD	Number of LORegion descriptors supported by the PE. This is an 8-bit binary number. **0b00000100** Four LOR descriptors are supported	0x04
[15:8]	**RES0**	Reserved	**RES0**
[7:0]	LR	Number of LORegions supported by the PE. This is an 8-bit binary number. **Note:** If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease. **0b00000100** Four LORegions are supported	0x04

Access

MRS <Xt>, LORID_EL1

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

Accessibility

MRS <Xt>, LORID_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = LORID_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = LORID_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = LORID_EL1;
```

A.1.6 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-6: AArch64_imp_cpuactlr_el1 bit assignments

Table A-22: IMP_CPUACTLR_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_0

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

MSR S3_0_C15_C1_0, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR_EL1;
```

MSR S3_0_C15_C1_0, <Xt>

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

```
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL1 = X[t, 64];
```

A.1.7 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-7: AArch64_imp_cpuactlr2_el1 bit assignments

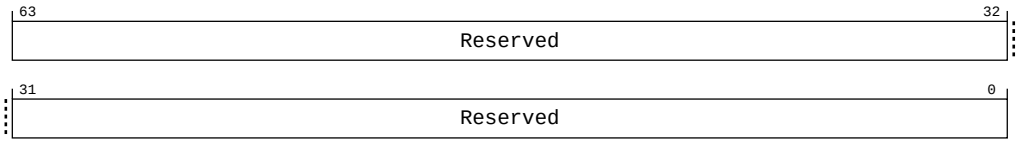


Table A-25: IMP_CPUACTLR2_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_1

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

MSR S3_0_C15_C1_1, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR2_EL1;
```

MSR S3_0_C15_C1_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR2_EL1 = X[t, 64];
```

A.1.8 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register 3 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-8: AArch64_imp_cpuactlr3_el1 bit assignments

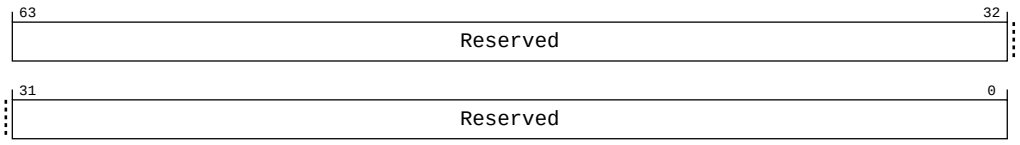


Table A-28: IMP_CPUACTLR3_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_2

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

MSR S3_0_C15_C1_2, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR3_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR3_EL1;
```

MSR S3_0_C15_C1_2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR3_EL1 = X[t, 64];
```

A.1.9 IMP_CPUACTLR4_EL1, CPU Auxiliary Control Register 4 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-9: AArch64_imp_cpuactlr4_el1 bit assignments

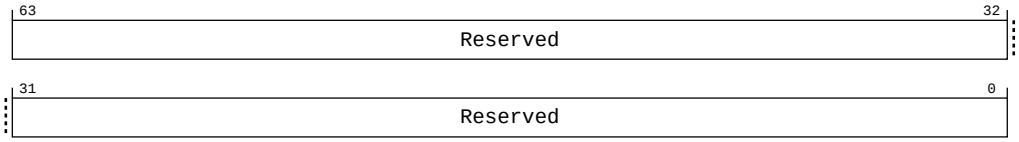


Table A-31: IMP_CPUACTLR4_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C1_3

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

MSR S3_0_C15_C1_3, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR4_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR4_EL1;
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = IMP_CPUACTLR4_EL1;
```

MSR S3_0_C15_C1_3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR4_EL1 = X[t, 64];
```

A.1.10 IMP_CPUECTLR_EL1, CPU Extended Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-10: AArch64_imp_cpuectlr_el1 bit assignments

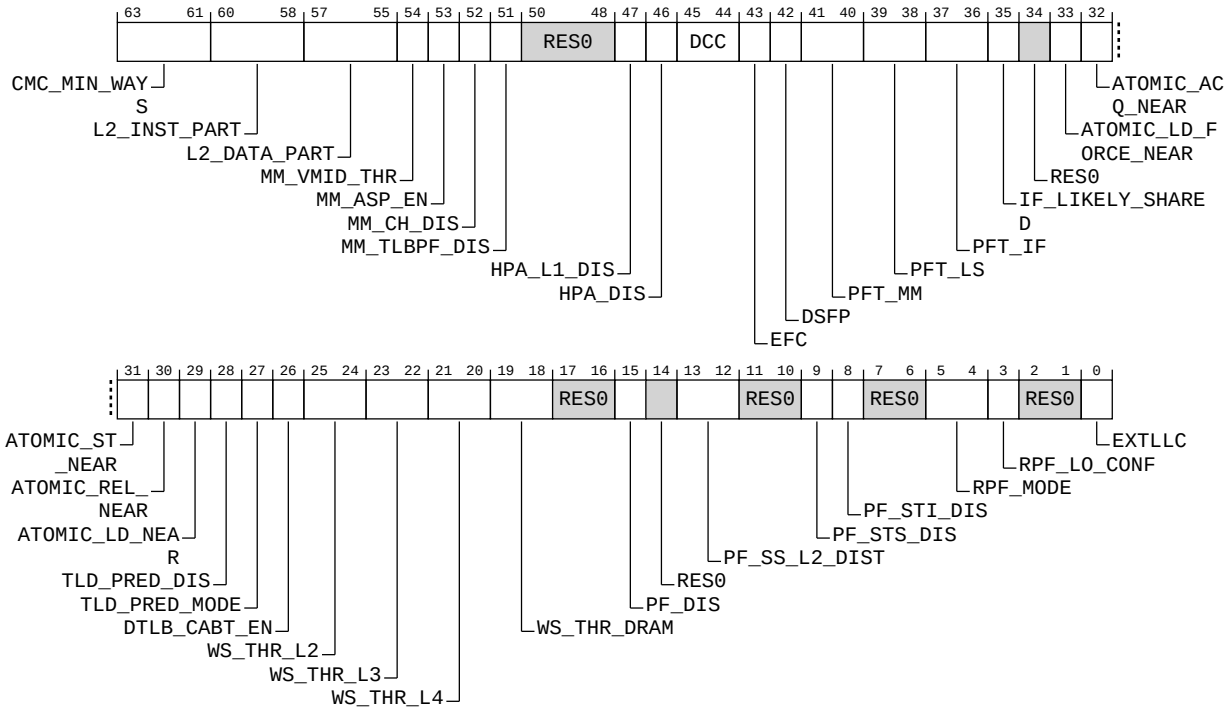


Table A-34: IMP_CPUECTLR_EL1 bit descriptions

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	xxx

Bits	Name	Description	Reset
[60:58]	L2_INST_PART	Partition the L2 cache for Instruction. The possible values are: **0b000** No ways reserved for instructions. This is the reset value **0b001** Reserve 1 way for instruction. Only instruction fetches can allocate way 7 **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	xxx
[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	xxx
[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	x

Bits	Name	Description	Reset
[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.	x
[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.	x
[51]	MM_TLBPF_DIS	Disables TLB prefetcher. The possible values are: **0b0** Enables TLB prefetcher. This is the default value. **0b1** Disables TLB prefetcher.	x
[50:48]	**RES0**	Reserved	**RES0**
[47]	HPA_L1_DIS	Disables hardware page aggregation in L1 TLBs. The possible values are: **0b0** Enables hardware page aggregation in L1 TLBs. This is the default value. **0b1** Disables hardware page aggregation in L1 TLBs.	x
[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.	x
[45:44]	DCC	Controls whether evictions of clean cache-lines send data on the CHI interface. Set this based on whether there is a cache on the path to memory. The possible values are: **0b00** Disables sending data when clean cache-lines are evicted. **0b01** Enables sending WriteEvictFull transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. **0b10** Enables sending WriteEvictOrEvict transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. 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.	xx

Bits	Name	Description	Reset
[43]	EFC	Eviction Flush Control (EFC). Controls whether hardware cache flushes and DC CISW instruction send data when evicting clan and dirty cache line. If it is known that data is likely to be used soon by another core, setting this bit can improve system performance. The possible values are: **0b0** Disables cache allocation in downstream caches when hardware cache flushes or DC CISW instructions evict a clean or cache line. Downstream Snoop Filter Present (DFSP) controls the sending of Evict transactions **0b1** Enables cache allocation in downstream caches when hardware cache flushes or DC CISW instructions evict a clean or dirty cache line. Downstream Cache Control (DCC) controls the sending of data. DSFP controls the sending of Evict transactions.	x
[42]	DSFP	Downstream Snoop Filter Present (DSFP). Enables sending Evict transactions on the CHI interface when clean lines are evicted without data. You must enable this if there is at least one snoop filter in the path to memory **0b0** Disables sending Evict transactions when clean cachelines are evicted without data **0b1** Enables sending of Evict transaction when clean cachelines are evicted without data.	x
[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.	xx
[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.	xx

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.	xx
[35]	IF_LIKELY_SHARED	Instruction fetch Shared state control. The possible values are: **0b0** Instruction fetch requests do not assert TXREQ LikelyShared. This is the reset value. **0b1** Instruction fetch requests assert TXREQ LikelyShared and request a SharedClean copy of data.	x
[34]	**RES0**	Reserved	**RES0**
[33]	ATOMIC_LD_FORCE_NEAR	A load atomic (including SWP & CAS) instruction to WB memory will be performed near. The possible values are: **0b0** Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **0b1** Load-atomic will be performed near by bringing the line into the L1D Cache. This is the default value.	x
[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.	x
[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.	x

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.	x
[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.	x
[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.	x
[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.	x
[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.	x
[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.	xx

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.	xx
[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.	xx
[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.	xx
[17:16]	**RES0**	Reserved	**RES0**
[15]	PF_DIS	Disables hardware prefetching. The possible values are: **0b0** Enables hardware prefetching. This is the default value. **0b1** Disables hardware prefetching.	x
[14]	**RES0**	Reserved	**RES0**

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.	xx
[11:10]	**RES0**	Reserved	**RES0**
[9]	PF_STS_DIS	Disable store-stride prefetches. The possible values are: **0b0** Enables store prefetching. This is the default value. **0b1** Disables store prefetching.	x
[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.	x
[7:6]	**RES0**	Reserved	**RES0**
[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.	xx
[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.	x
[2:1]	**RES0**	Reserved	**RES0**

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.	x

Access

MRS <Xt>, S3_0_C15_C1_4

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

MSR S3_0_C15_C1_4, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUECTLR_EL1;
```

MSR S3_0_C15_C1_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        IMP_CPUECTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR_EL1 = X[t, 64];
```

A.1.11 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-11: AArch64_imp_cpuectlr2_el1 bit assignments

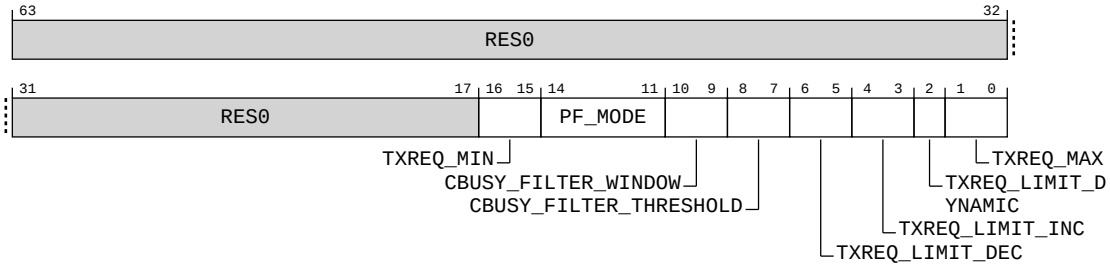


Table A-37: IMP_CPUECTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:17]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[16:15]	TXREQ_MIN	Minimum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are: **0b00** 1/4 of L2 TQ size - This is the default value **0b01** 1/8 of L2 TQ size **0b10** 1/16 of L2 TQ size **0b11** 1/32 of L2 TQ size	xx

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	xxxx

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	xx
[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	xx
[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	xx
[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	xx

Bits	Name	Description	Reset
[2]	TXREQ_LIMIT_DYNAMIC	Selects static or dynamic control of TXREQ limit. Dynamic TXREQ limit will adjust based on CBusy responses on RXDAT and RXRSP in the range of the static limit selected by CPUECTLR2_EL1[1:0] and 1/4 of the L2 TQ SIZE. The possible values are: **0b0** maximum number of TXREQ transactions statically set by CPUECTLR2_EL1[1:0] - This is the default value. **0b1** maximum number of TXREQ transactions dynamically controlled	x
[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	xx

Access

MRS <Xt>, S3_0_C15_C1_5

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

MSR S3_0_C15_C1_5, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUECTLR2_EL1;
```

MSR S3_0_C15_C1_5, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR2_EL1 = X[t, 64];
```

A.1.12 IMP_CPUL2DIRTYLNCT_EL1, CPU L2 Dirty Line Count Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0000 0000 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-12: AArch64_imp_cpul2dirtylnct_el1 bit assignments

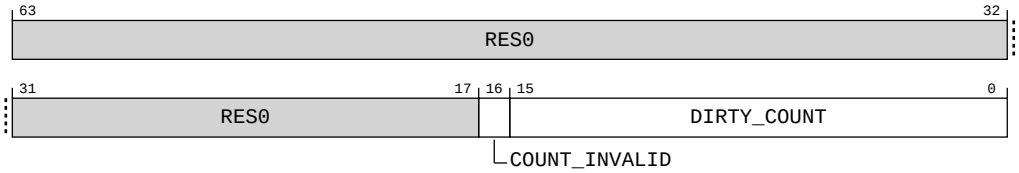


Table A-40: IMP_CPUL2DIRTYLNCT_EL1 bit descriptions

Bits	Name	Description	Reset
[63:17]	**RES0**	Reserved	**RES0**
[16]	COUNT_INVALID	Indicates the dirty count is invalid. Reset value is 'b0	0b0
[15:0]	DIRTY_COUNT	Number of dirty lines in the L2. Reset value is 'h0000	0x0000

Access

MRS <Xt>, S3_0_C15_C2_5

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b101

Accessibility

MRS <Xt>, S3_0_C15_C2_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUL2DIRTYLNCT_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUL2DIRTYLNCT_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUL2DIRTYLNCT_EL1;
```

A.1.13 IMP_CPUPWRCTLR_EL1, CPU Power Control Register

This register controls various power aspects of the core.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0000 xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-13: AArch64_imp_cpupwrctlr_el1 bit assignments

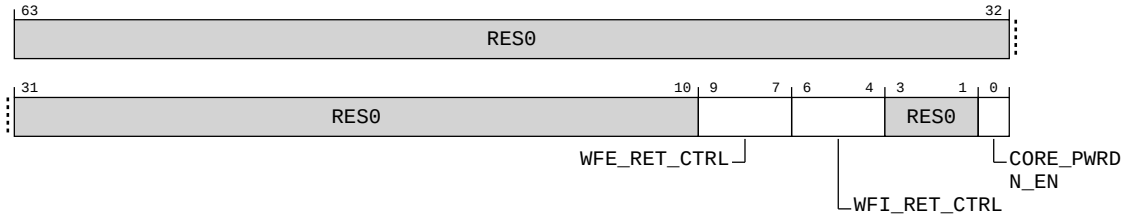


Table A-42: IMP_CPUPWRCTLR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[9:7]	WFE_RET_CTRL	Wait for Event retention control. The possible values are: **0b000** Dynamic retention is disabled. **0b001** 2 system counter ticks are required before retention entry. **0b010** 8 system counter ticks are required before retention entry. **0b011** 32 system counter ticks are required before retention entry. **0b100** 64 system counter ticks are required before retention entry. **0b101** 128 system counter ticks are required before retention entry. **0b110** 256 system counter ticks are required before retention entry. **0b111** 512 system counter ticks are required before retention entry.	0b000
[6:4]	WFI_RET_CTRL	Wait for Interrupt retention control. The possible values are: **0b000** Dynamic retention is disabled. **0b001** 2 system counter ticks are required before retention entry. **0b010** 8 system counter ticks are required before retention entry. **0b011** 32 system counter ticks are required before retention entry. **0b100** 64 system counter ticks are required before retention entry. **0b101** 128 system counter ticks are required before retention entry. **0b110** 256 system counter ticks are required before retention entry. **0b111** 512 system counter ticks are required before retention entry.	0b000
[3:1]	**RES0**	Reserved	**RES0**
[0]	CORE_PWRDN_EN	Indicates to the power controller if the CPU wants to power down when it enters WFE/WFI state. The possible values are: **0b0** CPU does not want to power down when it enters WFE/WFI state. **0b1** CPU wants to power down when it enters WFE/WFI state.	0b0

Access

MRS <Xt>, S3_0_C15_C2_7

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

MSR S3_0_C15_C2_7, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C2_7

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPWRCTLR_EL1;
```

MSR S3_0_C15_C2_7, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.PWREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUPWRCTLR_EL1 = X[t, 64];
```

A.1.14 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register (EL1)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-14: AArch64_imp_atcr_el1 bit assignments

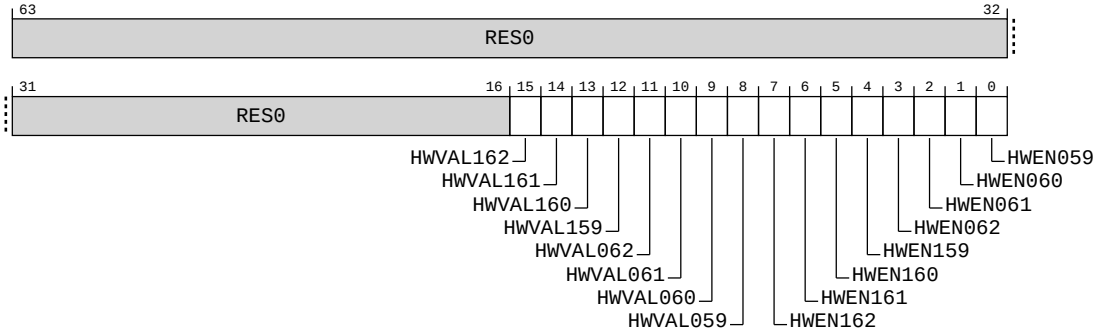


Table A-45: IMP_ATCR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0

Access

MRS <Xt>, S3_0_C15_C7_0

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

MSR S3_0_C15_C7_0, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C7_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_ATCR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_ATCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL1;
```

MSR S3_0_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
```

```
    IMP_ATCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t, 64];
```

A.1.15 IMP_CPUACTLR5_EL1, CPU Auxiliary Control Register 5 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-15: AArch64_imp_cpuactlr5_el1 bit assignments

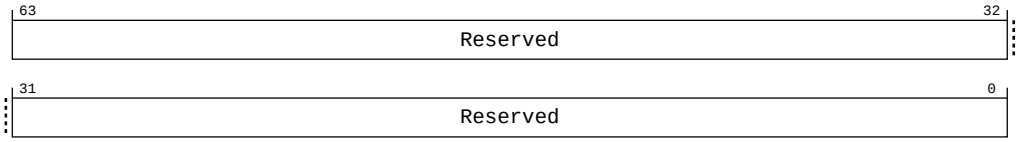


Table A-48: IMP_CPUACTLR5_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_0

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

MSR S3_0_C15_C8_0, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C8_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR5_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR5_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR5_EL1;
```

MSR S3_0_C15_C8_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR5_EL1 = X[t, 64];
```

A.1.16 IMP_CPUACTLR6_EL1, CPU Auxiliary Control Register 6 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-16: AArch64_imp_cpuactlr6_el1 bit assignments

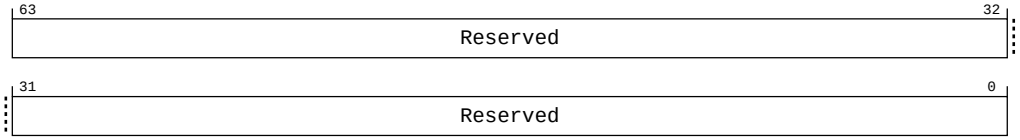


Table A-51: IMP_CPUACTLR6_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b001

MSR S3_0_C15_C8_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b001

Accessibility

MRS <Xt>, S3_0_C15_C8_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR6_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR6_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR6_EL1;
```

MSR S3_0_C15_C8_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR6_EL1 = X[t, 64];
```

A.1.17 IMP_CPUACTLR7_EL1, CPU Auxiliary Control Register 7 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-17: AArch64_imp_cpuactlr7_el1 bit assignments

Table A-54: IMP_CPUACTLR7_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_2

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b010

MSR S3_0_C15_C8_2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b010

Accessibility

MRS <Xt>, S3_0_C15_C8_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR7_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR7_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR7_EL1;
```

MSR S3_0_C15_C8_2, <Xt>

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

```
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR7_EL1 = X[t, 64];
```

A.1.18 IMP_CPUACTLR8_EL1, CPU Auxiliary Control Register 8 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-18: AArch64_imp_cpuactlr8_el1 bit assignments

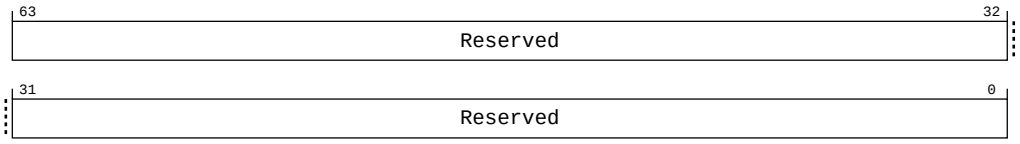


Table A-57: IMP_CPUACTLR8_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_5

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b101

MSR S3_0_C15_C8_5, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b1000	0b101

Accessibility

MRS <Xt>, S3_0_C15_C8_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR8_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR8_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR8_EL1;
```

MSR S3_0_C15_C8_5, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR8_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR8_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR8_EL1 = X[t, 64];
```

A.1.19 IMP_CPUACTLR9_EL1, CPU Auxiliary Control Register 9 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-19: AArch64_imp_cpuactlr9_el1 bit assignments

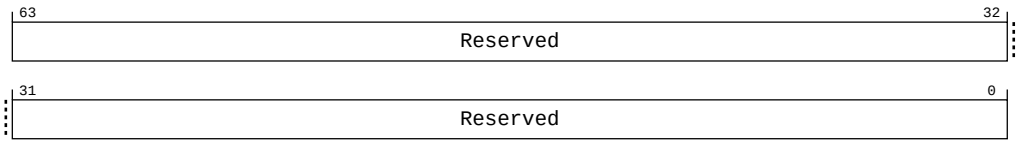


Table A-60: IMP_CPUACTLR9_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C8_6

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

MSR S3_0_C15_C8_6, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C8_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUACTLR9_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR9_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR9_EL1;
```

MSR S3_0_C15_C8_6, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR9_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR9_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR9_EL1 = X[t, 64];
```

A.1.20 AIDR_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-20: AArch64_aidr_el1 bit assignments

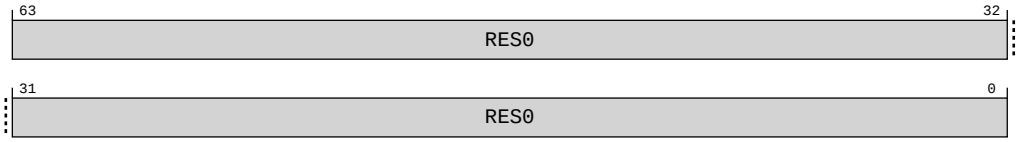


Table A-63: AIDR_EL1 bit descriptions

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

Access

MRS <Xt>, AIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, AIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AIDR_EL1;
```

A.1.21 FPCR, Floating-point Control Register

Controls floating-point behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-21: AArch64_fpcr bit assignments

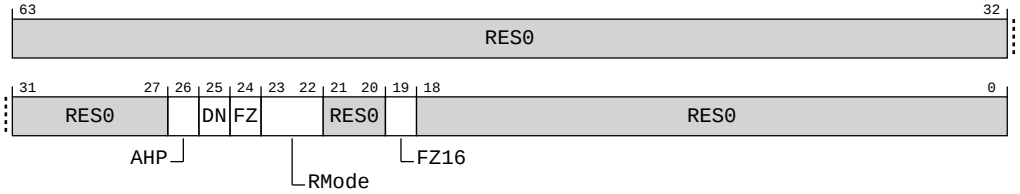


Table A-65: FPCR bit descriptions

Bits	Name	Description	Reset
[63:27]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[26]	AHP	Alternative half-precision control bit. **0b0** IEEE half-precision format selected. **0b1** Alternative half-precision format selected. This bit is used only for conversions between half-precision floating-point and other floating-point formats. The data-processing instructions added as part of the FEAT_FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.	x
[25]	DN	Default NaN use for NaN propagation. **0b0** NaN operands propagate through to the output of a floating-point operation. **0b1** Any operation involving one or more NaNs returns the Default NaN. This bit has no effect on the output of FABS, FMAX*, FMIN*, and FNEG instructions, and a default NaN is never returned as a result of these instructions. The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.	x
[24]	FZ	Flushing denormalized numbers to zero control bit. **0b0** Flushing denormalized numbers to zero disabled. **0b1** Flushing denormalized numbers to zero enabled. For more information, see *Flushing denormalized numbers to zero* in the *Arm® Architecture Reference Manual for A-profile architecture* and the pseudocode of the floating-point instructions.	x
[23:22]	RMode	Rounding Mode control field. **0b00** Round to Nearest (RN) mode. **0b01** Round towards Plus Infinity (RP) mode. **0b10** Round towards Minus Infinity (RM) mode. **0b11** Round towards Zero (RZ) mode. The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.	xx
[21:20]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[19]	FZ16	Flushing denormalized numbers to zero control bit on half-precision data-processing instructions. **0b0** For some instructions, this bit disables flushing to zero of inputs and outputs that are half-precision denormalized numbers. **0b1** Flushing denormalized numbers to zero enabled. The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations. For more information, see *Flushing denormalized numbers to zero* in the *Arm® Architecture Reference Manual for A-profile architecture* and the pseudocode of the floating-point instructions.	x
[18:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, FPCR

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

MSR FPCR, <Xt>

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

Accessibility

MRS <Xt>, FPCR

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = 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
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
```

MSR FPCR, <Xt>

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
```

```
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
```

A.1.22 FPSR, Floating-point Status Register

Provides floating-point system status information.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-22: AArch64_fpsr bit assignments

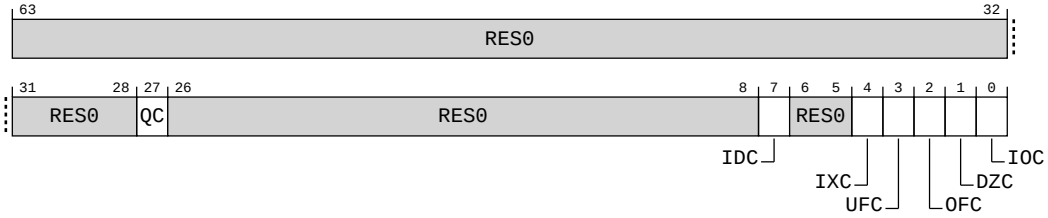


Table A-68: FPSR bit descriptions

Bits	Name	Description	Reset
[63:28]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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.	x
[26:8]	**RES0**	Reserved	**RES0**
[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 scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.IDE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.IDE is 0.	x
[6:5]	**RES0**	Reserved	**RES0**
[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 scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.IXE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.IXE is 0. The criteria for the Inexact floating-point exception to occur are affected by whether denormalized numbers are flushed to zero and by the value of the AArch64-FPCR.AH bit. For more information, see *Floating-point exceptions and exception traps* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[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 scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.UFE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.UFE is 0 or if flushing denormalized numbers to zero is enabled. The criteria for the Underflow floating-point exception to occur are affected by whether denormalized numbers are flushed to zero and by the value of the AArch64-FPCR.AH bit. For more information, see *Floating-point exceptions and exception traps* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[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 scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.OFE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.OFE is 0.	x
[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 scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.DZE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.DZE is 0.	x
[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 scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.IOE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.IOE is 0. The criteria for the Invalid Operation floating-point exception to occur are affected by the value of the AArch64-FPCR.AH bit. For more information, see *Floating-point exceptions and exception traps* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x

Access

MRS <Xt>, FPSR

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b001

MSR FPSR, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b001

Accessibility

MRS <Xt>, FPSR

```
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
```

MSR FPSR, <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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
```

A.1.23 ACTLR_EL2, Auxiliary Control Register (EL2)

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


Note

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

Configurations

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

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

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0xxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-23: AArch64_actlr_el2 bit assignments

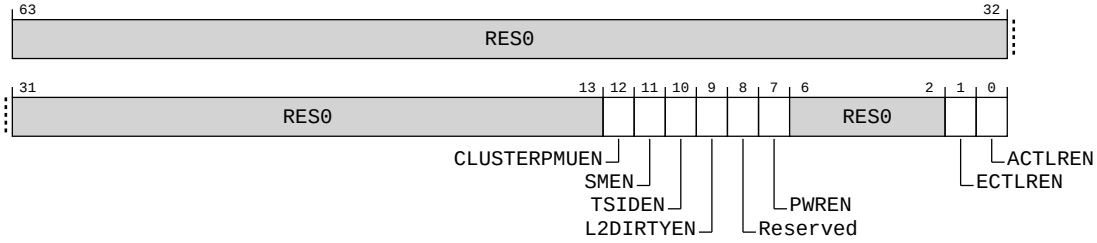


Table A-71: ACTLR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:13]	**RES0**	Reserved	**RES0**
[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.	0b0

Bits	Name	Description	Reset
[11]	SMEN	Scheme Management Registers enable. The possible values are: **0b0** Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL1. This is the reset value. **0b1** Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL1 if they are write-accessible from EL2.	0b0
[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	0b0
[9]	L2DIRTYEN	L2 Dirty Line Count Register enable. The possible values are: **0b0** Register CPUL2DIRTYLNCT_EL1 is not read-accessible in EL1. This is the reset value. **0b1** Register CPUL2DIRTYLNCT_EL1 is read-accessible in EL1.	0b0
[8]	Reserved	Reserved **0b0** Reserved **0b1** Reserved	0b0
[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	0b0
[6:2]	**RES0**	Reserved	**RES0**
[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

Bits	Name	Description	Reset
[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

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b001

MSR ACTLR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b001

Accessibility

MRS <Xt>, ACTLR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ACTLR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL2;
```

MSR ACTLR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    ACTLR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ACTLR_EL2 = X[t, 64];
```

A.1.24 HACR_EL2, Hypervisor Auxiliary Control Register

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

Note

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

Configurations

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

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

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-24: AArch64_hacr_el2 bit assignments

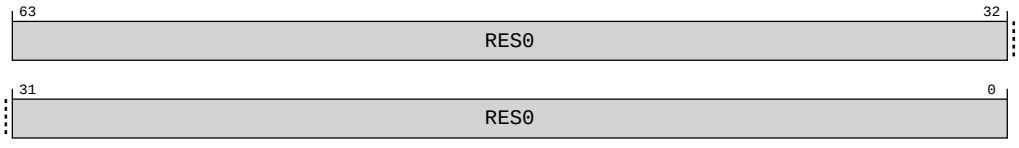


Table A-74: HACR_EL2 bit descriptions

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

Access

MRS <Xt>, HACR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b111

MSR HACR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b111

Accessibility

MRS <Xt>, HACR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HACR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HACR_EL2;
```

MSR HACR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    HACR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HACR_EL2 = X[t, 64];
```

A.1.25 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL2.

Configurations

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

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

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-25: AArch64_afsr0_el2 bit assignments

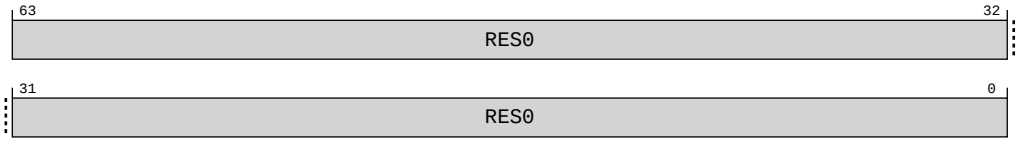


Table A-77: AFSR0_EL2 bit descriptions

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

Access

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

MRS <Xt>, AFSR0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

MSR AFSR0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

MRS <Xt>, AFSR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

MSR AFSR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b000

Accessibility

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

MRS <Xt>, AFSR0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL2;
```

MSR AFSR0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR0_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL2 = X[t, 64];
```

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR0_EL2;
    else
        X[t, 64] = AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL2 = X[t, 64];
```

```
    else
        AFSR0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t, 64];
```

A.1.26 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL2.

Configurations

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

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

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-26: AArch64_afsr1_el2 bit assignments

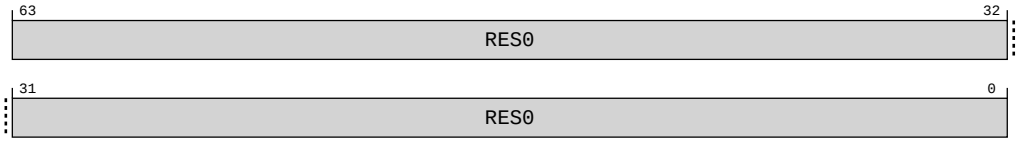


Table A-82: AFSR1_EL2 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR1_EL2 or AFSR1_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

MRS <Xt>, AFSR1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b001

MSR AFSR1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b001

MRS <Xt>, AFSR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

MSR AFSR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0001	0b001

Accessibility

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR1_EL2 or AFSR1_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.
MRS <Xt>, AFSR1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR1_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL2;
```

MSR AFSR1_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR1_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL2 = X[t, 64];
```

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AFSR1_EL2;
    else
        X[t, 64] = AFSR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL1;
```

MSR AFSR1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL2 = X[t, 64];
    else
        AFSR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL1 = X[t, 64];
```

A.1.27 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

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

Configurations

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

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

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

AMAIR_EL2 is permitted to be cached in a TLB.

Figure A-27: AArch64_amair_el2 bit assignments

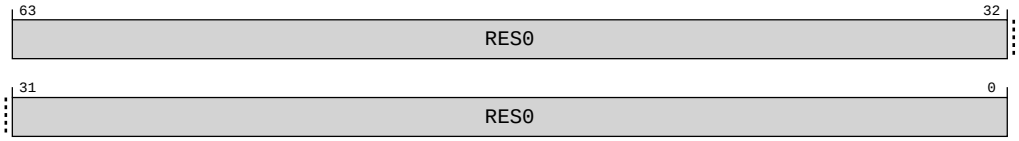


Table A-87: AMAIR_EL2 bit descriptions

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

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AMAIR_EL2 or AMAIR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

MRS <Xt>, AMAIR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b000

MSR AMAIR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0011	0b000

MRS <Xt>, AMAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0011	0b000

Accessibility

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic AMAIR_EL2 or AMAIR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

MRS <Xt>, AMAIR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AMAIR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL2;
```

MSR AMAIR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AMAIR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL2 = X[t, 64];
```

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = AMAIR_EL2;
    else
        X[t, 64] = AMAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL1;
```

MSR AMAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL2 = X[t, 64];
```

```
    else
        AMAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t, 64];
```

A.1.28 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-28: AArch64_imp_atcr_el2 bit assignments

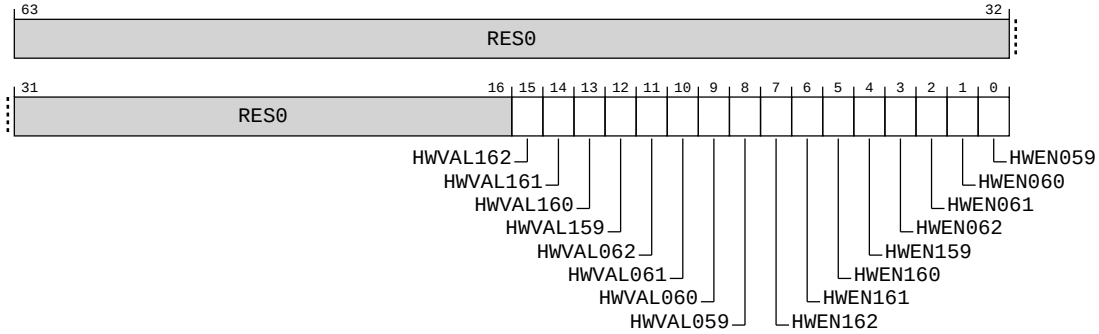


Table A-92: IMP_ATCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using TTBR1_EL2 if HWEN162 is set.	0b0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using TTBR1_EL2 if HWEN161 is set.	0b0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using TTBR1_EL2 if HWEN160 is set.	0b0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using TTBR1_EL2 if HWEN159 is set.	0b0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2 if HWEN062 is set.	0b0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2 if HWEN059 is set.	0b0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0

Access

MRS <Xt>, S3_4_C15_C7_0

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b000

MSR S3_4_C15_C7_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_4_C15_C7_0

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

```
    X[t, 64] = IMP_ATCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL2;
```

MSR S3_4_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL2 = X[t, 64];
```

A.1.29 IMP_AVTCR_EL2, CPU Virtualization Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-29: AArch64_imp_avtcr_el2 bit assignments

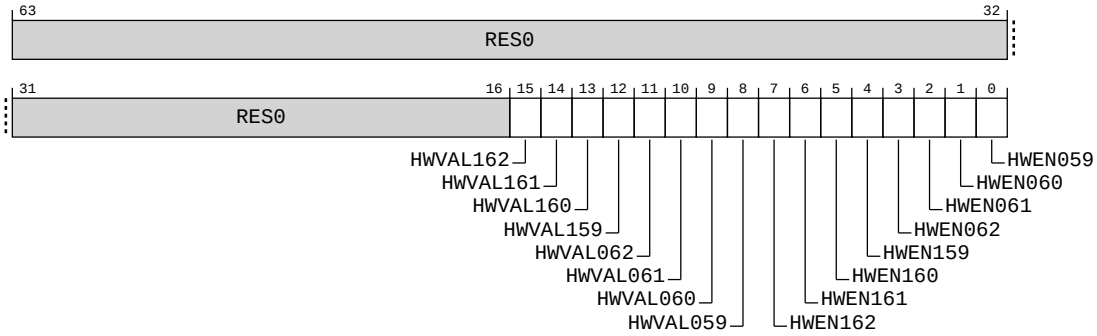


Table A-95: IMP_AVTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN162 is set.	0b0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN161 is set.	0b0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN160 is set.	0b0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN159 is set.	0b0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2 if HWEN062 is set.	0b0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2 if HWEN059 is set.	0b0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0

Access

MRS <Xt>, S3_4_C15_C7_1

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

MSR S3_4_C15_C7_1, <Xt>

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

Accessibility

MRS <Xt>, S3_4_C15_C7_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_AVTCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_AVTCR_EL2;
```

MSR S3_4_C15_C7_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_AVTCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_AVTCR_EL2 = X[t, 64];
```

A.1.30 ACTLR_EL3, Auxiliary Control Register (EL3)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0xxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-30: AArch64_actlr_el3 bit assignments

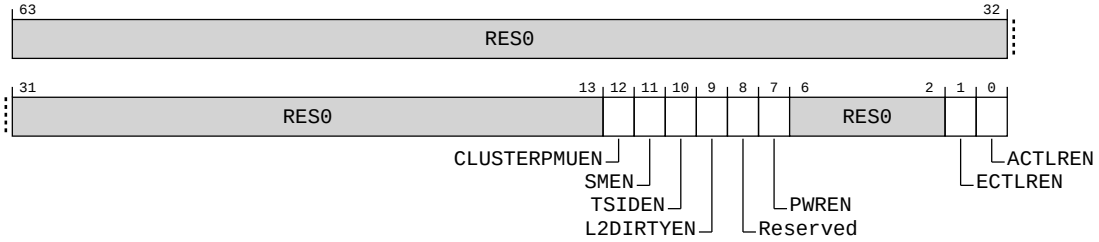


Table A-98: ACTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:13]	**RES0**	Reserved	**RES0**
[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.	0b0
[11]	SMEN	Scheme Management Registers enable. The possible values are: **0b0** Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL2 and EL1. This is the reset value. **0b1** Registers CLUSTERACPSID, CLUSTERCFR2, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL2 and EL1 if they are write-accessible from EL2.	0b0
[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	0b0

Bits	Name	Description	Reset
[9]	L2DIRTYEN	L2 Dirty Line Count Register enable. The possible values are: **0b0** Register CPUL2DIRTYLNCT_EL1 is not read-accessible in EL2 and EL1. This is the reset value. **0b1** Register CPUL2DIRTYLNCT_EL1 is read-accessible in EL2 and EL1.	0b0
[8]	Reserved	Reserved **0b0** Reserved **0b1** Reserved	0b0
[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	0b0
[6:2]	**RES0**	Reserved	**RES0**
[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
[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

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b001

MSR ACTLR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0000	0b001

Accessibility

MRS <Xt>, ACTLR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL3;
```

MSR ACTLR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ACTLR_EL3 = X[t, 64];
```

A.1.31 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-31: AArch64_afsr0_el3 bit assignments

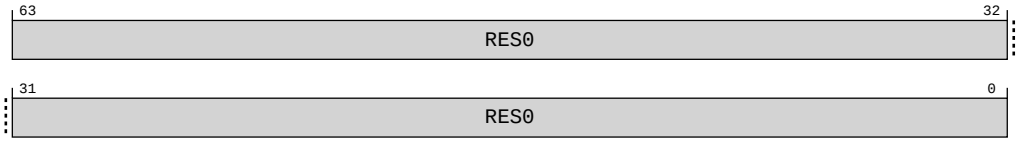


Table A-101: AFSR0_EL3 bit descriptions

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

Access

MRS <Xt>, AFSR0_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b000

MSR AFSR0_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b000

Accessibility

MRS <Xt>, AFSR0_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR0_EL3;
```

MSR AFSR0_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AFSR0_EL3 = X[t, 64];
```

A.1.32 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-32: AArch64_afsr1_el3 bit assignments

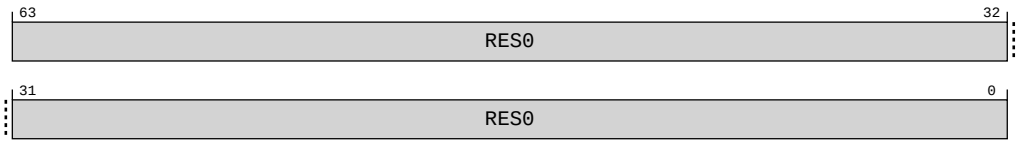


Table A-104: AFSR1_EL3 bit descriptions

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

Access

MRS <Xt>, AFSR1_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b001

MSR AFSR1_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0101	0b0001	0b001

Accessibility

MRS <Xt>, AFSR1_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AFSR1_EL3;
```

MSR AFSR1_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AFSR1_EL3 = X[t, 64];
```

A.1.33 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

AMAIR_EL3 is permitted to be cached in a TLB.

Figure A-33: AArch64_amair_el3 bit assignments

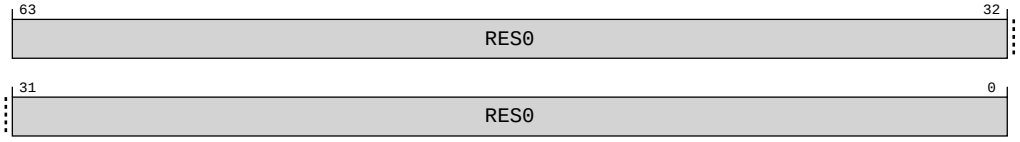


Table A-107: AMAIR_EL3 bit descriptions

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

Access

MRS <Xt>, AMAIR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0011	0b000

MSR AMAIR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1010	0b0011	0b000

Accessibility

MRS <Xt>, AMAIR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMAIR_EL3;
```

MSR AMAIR_EL3, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AMAIR_EL3 = X[t, 64];
```

A.1.34 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.
- In a AArch64 only implementation it is IMPLEMENTATION DEFINED whether the register is implemented.

Otherwise, direct accesses to RMR_EL3 are UNDEFINED.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx01

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-34: AArch64_rmr_el3 bit assignments

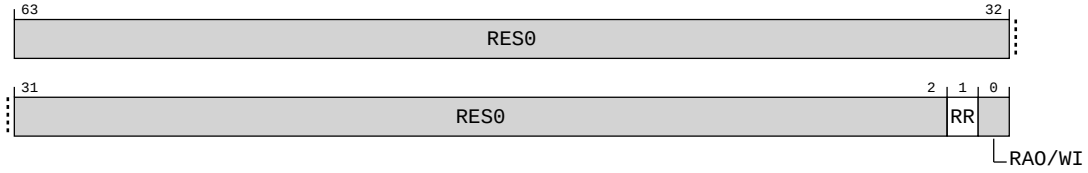


Table A-110: RMR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:2]	**RES0**	Reserved	**RES0**
[1]	RR	Reset Request. Setting this bit to 1 requests a Warm reset.	0b0
[0]	**RAO/WI**	Reserved	**RAO/WI**

Access

MRS <Xt>, RMR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b0000	0b010

MSR RMR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1100	0b0000	0b010

Accessibility

MRS <Xt>, RMR_EL3

```
if PSTATE.EL == EL3 then
    X[t, 64] = RMR_EL3;
else
    UNDEFINED;
```

MSR RMR_EL3, <Xt>

```
if PSTATE.EL == EL3 then
    RMR_EL3 = X[t, 64];
else
    UNDEFINED;
```

A.1.35 IMP_CPUL2SDIRTYLNCT_EL3, CPU L2 Secure Dirty Line Count Register

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0000 0000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-35: AArch64_imp_cpul2sdirtylnct_el3 bit assignments

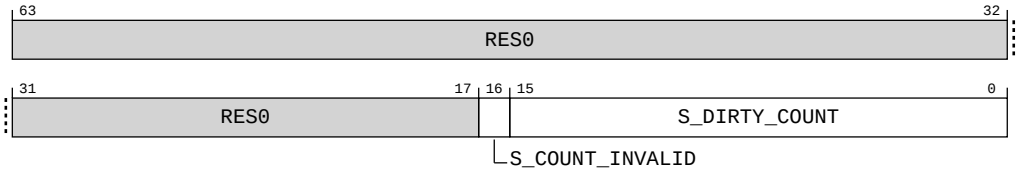


Table A-113: IMP_CPUL2SDIRTYLNCT_EL3 bit descriptions

Bits	Name	Description	Reset
[63:17]	RES0	Reserved	RES0
[16]	S_COUNT_INVALID	Indicates the secure dirty count is invalid. Reset value is 'b0	0b0
[15:0]	S_DIRTY_COUNT	Number of dirty secure lines in the L2. Reset value is 'h0000	0x0000

Access

MRS <Xt>, S3_6_C15_C2_3

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b011

Accessibility

MRS <Xt>, S3_6_C15_C2_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUL2SDIRTYLNCT_EL3;
```

A.1.36 IMP_CPUACTLR_EL3, CPU Auxiliary Control Register (EL3)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-36: AArch64_imp_cpuactlr_el3 bit assignments

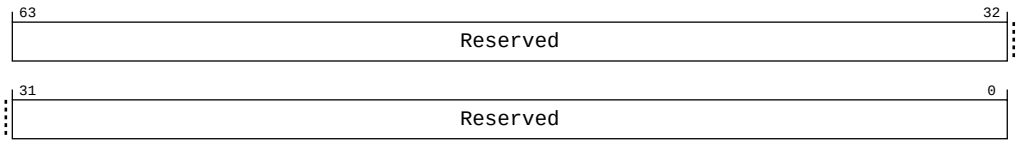


Table A-115: IMP_CPUACTLR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C4_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0100	0b000

MSR S3_6_C15_C4_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0100	0b000

Accessibility

MRS <Xt>, S3_6_C15_C4_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUACTLR_EL3;
```

MSR S3_6_C15_C4_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL3 = X[t, 64];
```

A.1.37 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

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-37: AArch64_imp_atcr_el3 bit assignments

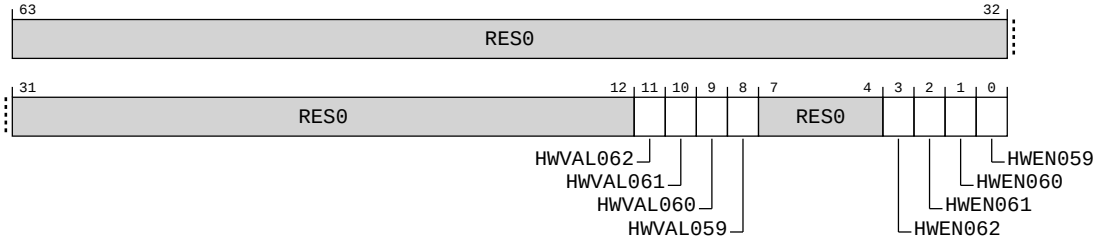


Table A-118: IMP_ATCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL3 if HWEN062 is set.	0b0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL3 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL3 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL3 if HWEN059 is set.	0b0
[7:4]	**RES0**	Reserved	**RES0**
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[3] will be 0 on page table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[2] will be 0 on page table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[1] will be 0 on page table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[0] will be 0 on page table walks.	0b0

Access

MRS <Xt>, S3_6_C15_C7_0

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

MSR S3_6_C15_C7_0, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C7_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL3;
```

MSR S3_6_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t, 64];
```

A.1.38 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

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-38: AArch64_imp_cpupselr_el3 bit assignments

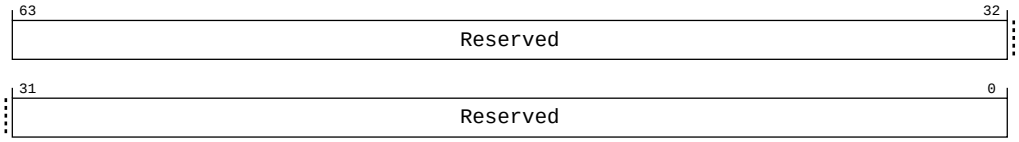


Table A-121: IMP_CPUPSELR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_0

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

MSR S3_6_C15_C8_0, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C8_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPSELR_EL3;
```

MSR S3_6_C15_C8_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPSELR_EL3 = X[t, 64];
```

A.1.39 IMP_CPUPCR_EL3, Selected Instruction Private Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-39: AArch64_imp_cpupcr_el3 bit assignments

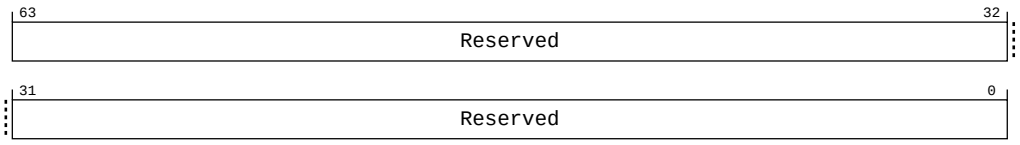


Table A-124: IMP_CPUPCR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_1

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

MSR S3_6_C15_C8_1, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C8_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPCR_EL3;
```

MSR S3_6_C15_C8_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPCR_EL3 = X[t, 64];
```

A.1.40 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

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-40: AArch64_imp_cpupor_el3 bit assignments

Table A-127: IMP_CPUPOR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_2

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

MSR S3_6_C15_C8_2, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C8_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPOR_EL3;
```

MSR S3_6_C15_C8_2, <Xt>

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

```
elsif PSTATE.EL == EL3 then
    IMP_CPUPOR_EL3 = X[t, 64];
```

A.1.41 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

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-41: AArch64_imp_cpupmr_el3 bit assignments

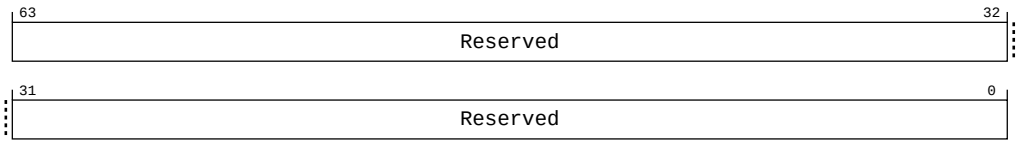


Table A-130: IMP_CPUPMR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_3

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

MSR S3_6_C15_C8_3, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C8_3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPMR_EL3;
```

MSR S3_6_C15_C8_3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR_EL3 = X[t, 64];
```

A.1.42 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

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-42: AArch64_imp_cpupor2_el3 bit assignments

Table A-133: IMP_CPUPOR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_4

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

MSR S3_6_C15_C8_4, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C8_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPOR2_EL3;
```

MSR S3_6_C15_C8_4, <Xt>

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

```
elsif PSTATE.EL == EL3 then
    IMP_CPUPOR2_EL3 = X[t, 64];
```

A.1.43 IMP_CPUPMR2_EL3, Selected Instruction Private Mask Register 2

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-43: AArch64_imp_cpupmr2_el3 bit assignments

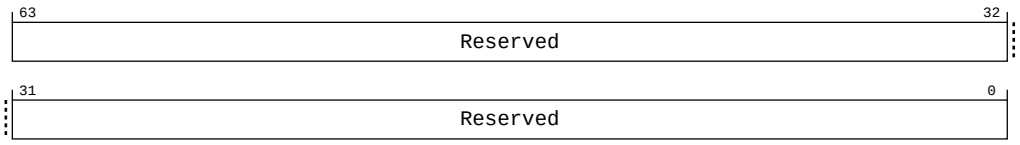


Table A-136: IMP_CPUPMR2_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_5

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

MSR S3_6_C15_C8_5, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C8_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPMR2_EL3;
```

MSR S3_6_C15_C8_5, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPMR2_EL3 = X[t, 64];
```

A.1.44 IMP_CPUPFR_EL3, Selected Instruction Private Flag Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-44: AArch64_imp_cpupfr_el3 bit assignments

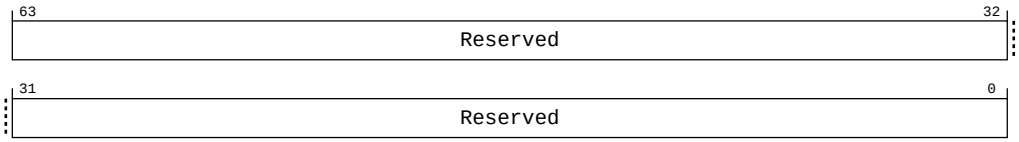


Table A-139: IMP_CPUPFR_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C8_6

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

MSR S3_6_C15_C8_6, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C8_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPFR_EL3;
```

MSR S3_6_C15_C8_6, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPFR_EL3 = X[t, 64];
```

A.2 AArch64 Debug registers summary

The summary table provides an overview of all Debug registers in the core. For more information about a register, click the register name in the table.

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

Table A-142: Debug registers summary

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

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

A.2.1 DBGBVR0_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register AArch64-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of AArch64-DBGBCR<n>_EL1.BT.

- When AArch64-DBGBCR<n>_EL1.BT is 0b000x, this register holds a virtual address.
- When AArch64-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of AArch64-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When AArch64-DBGBCR0_EL1.BT == '000x'

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

When AArch64-DBGBCR0_EL1.BT == '001x'

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

When AArch64-DBGBCR0_EL1.BT == '011x'

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

When AArch64-DBGBCR0_EL1.BT == '100x'

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

When AArch64-DBGBCR0_EL1.BT == '101x'

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

When AArch64-DBGBCR0_EL1.BT == '110x'

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

When AArch64-DBGBCR0_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-DBGBCR0_EL1.BT == '000'

Figure A-45: AArch64_dbgbvr0_el1 bit assignments

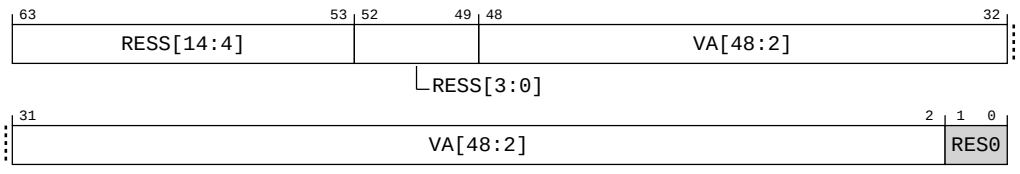


Table A-143: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • If the breakpoint is not context-aware, it is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR0_EL1.BT == '001'

Figure A-46: AArch64_dbgbvr0_el1 bit assignments

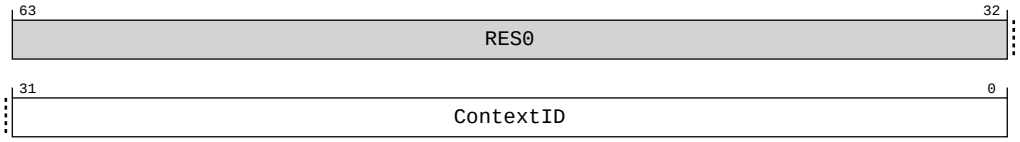


Table A-144: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR0_EL1.BT == '011'

Figure A-47: AArch64_dbgbvr0_el1 bit assignments

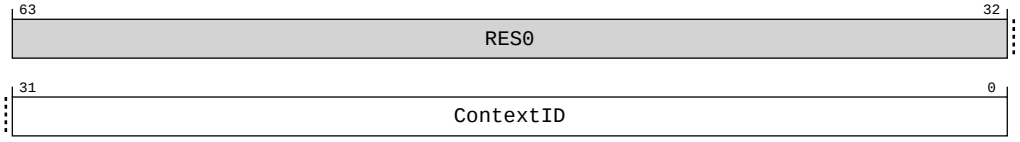


Table A-145: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR0_EL1.BT == '100'

Figure A-48: AArch64_dbgbvr0_el1 bit assignments

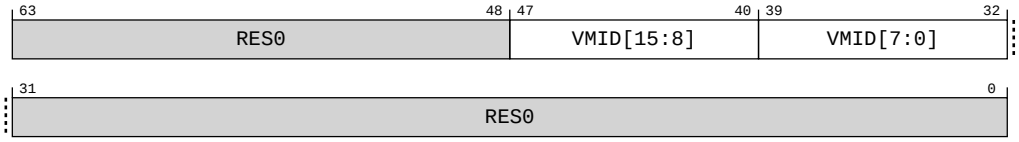


Table A-146: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR0_EL1.BT == '101'

Figure A-49: AArch64_dbgbvr0_el1 bit assignments

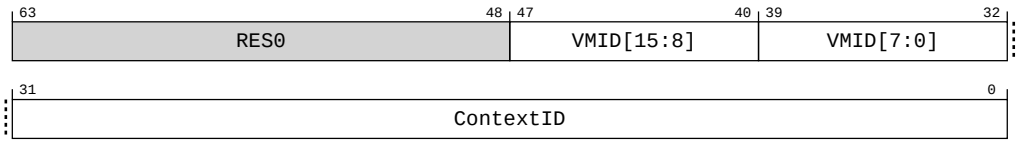


Table A-147: DBGBVR0_EL1 bit descriptions

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

Bits	Name	Description	Reset
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR0_EL1.BT == '110'

Figure A-50: AArch64_dbgbvr0_el1 bit assignments

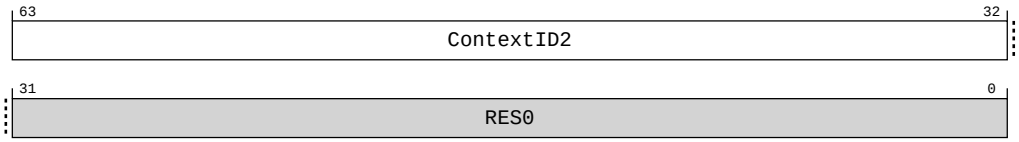


Table A-148: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR0_EL1.BT == '111'

Figure A-51: AArch64_dbgbvr0_el1 bit assignments

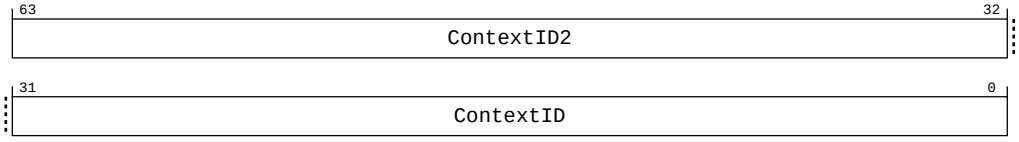


Table A-149: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

MRS <Xt>, DBGBVR0_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b100

MSR DBGBVR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b100

Accessibility

MRS <Xt>, DBGBVR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[0];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[0];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[0];
```

MSR DBGBVR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[0] = X[t, 64];
```

A.2.2 DBGBCR0_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register AArch64-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented, accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-52: AArch64_dbgbcr0_el1 bit assignments

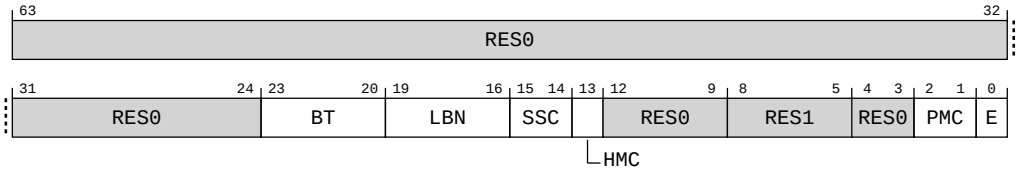


Table A-152: DBGBCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000, but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value **0b0011** As 0b0010, with linking enabled. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. AArch64-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGBCR<n>_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	xx
[0]	E	Enable breakpoint AArch64-DBGBVR<n>_EL1. **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

MRS <Xt>, DBGBCR0_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b101

MSR DBGBCR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b101

Accessibility

MRS <Xt>, DBGBCR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[0];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[0];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[0];
```

MSR DBGBCR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[0] = X[t, 64];
```

A.2.3 DBGWVR0_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register AArch64-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-53: AArch64_dbgwvr0_el1 bit assignments

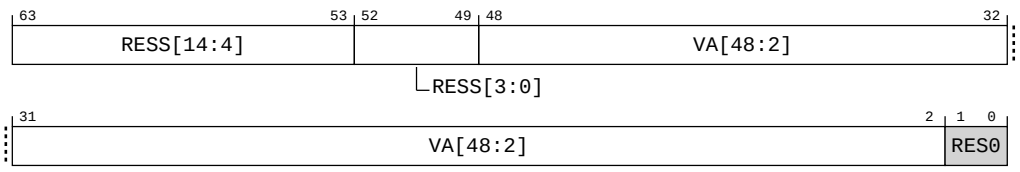


Table A-155: DBGWVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • It is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, DBGWVR0_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b110

MSR DBGWVR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b110

Accessibility

MRS <Xt>, DBGWVR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[0];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[0];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
```

```
    else
        X[t, 64] = DBGWVR_EL1[0];
```

MSR DBGWVR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[0] = X[t, 64];
```

A.2.4 DBGWCR0_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register AArch64-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-54: AArch64_dbgwcr0_el1 bit assignments

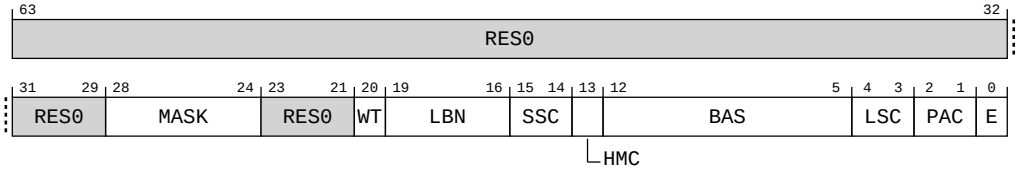


Table A-158: DBGWCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	`5{x}`
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	`x`
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	`xxxx`

Bits	Name	Description	Reset
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGWCR<n>_EL1.{SSC, HMC, PAC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by AArch64-DBGWVR<n>_EL1 is being watched. Table A-159: BAS description table 1 on page 271 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-160: BAS description table 2 on page 271 If AArch64-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>_EL1.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx

Bits	Name	Description	Reset
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table A-159: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table A-160: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

MRS <Xt>, DBGWCR0_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b111

MSR DBGWCR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, DBGWCR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[0];
```

```
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[0];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[0];
```

MSR DBGWCR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[0] = X[t, 64];
```

A.2.5 DBGBVR1_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register AArch64-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of AArch64-DBGBCR<n>_EL1.BT.

- When AArch64-DBGBCR<n>_EL1.BT is 0b000x, this register holds a virtual address.
- When AArch64-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.

- When AArch64-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of AArch64-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When AArch64-DBGBCR1_EL1.BT == '000x'

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

When AArch64-DBGBCR1_EL1.BT == '001x'

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

When AArch64-DBGBCR1_EL1.BT == '011x'

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

When AArch64-DBGBCR1_EL1.BT == '100x'

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

When AArch64-DBGBCR1_EL1.BT == '101x'

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

When AArch64-DBGBCR1_EL1.BT == '110x'

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

When AArch64-DBGBCR1_EL1.BT == '111x'

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-DBGBCR1_EL1.BT == '000'

Figure A-55: AArch64_dbgbvr1_el1 bit assignments

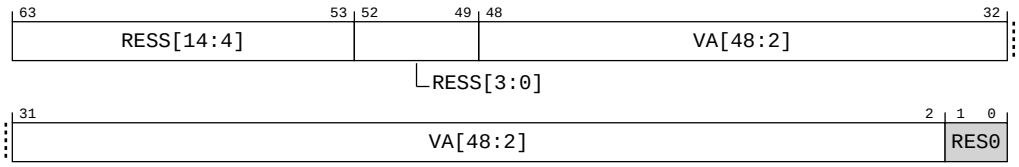


Table A-163: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • If the breakpoint is not context-aware, it is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR1_EL1.BT == '001'

Figure A-56: AArch64_dbgbvr1_el1 bit assignments

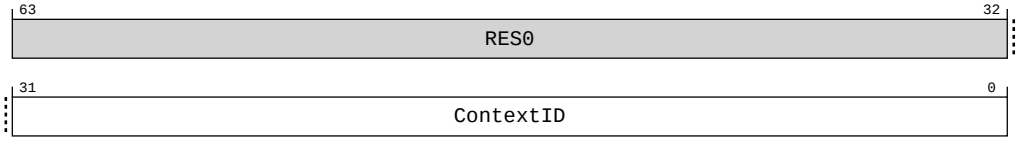


Table A-164: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR1_EL1.BT == '011'

Figure A-57: AArch64_dbgbvr1_el1 bit assignments

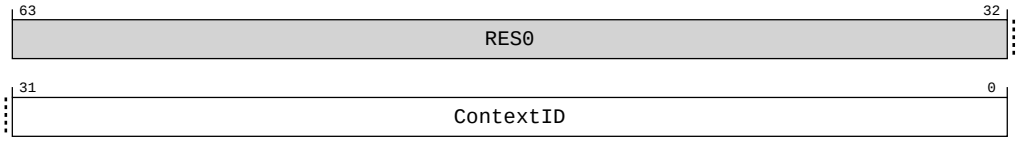


Table A-165: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR1_EL1.BT == '100'

Figure A-58: AArch64_dbgbvr1_el1 bit assignments

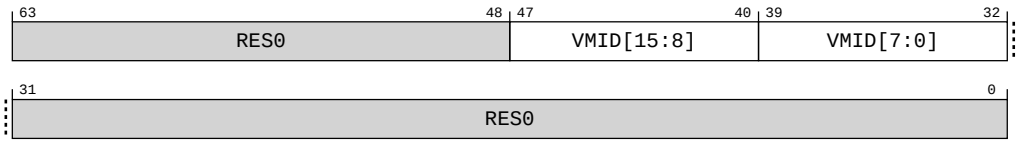


Table A-166: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR1_EL1.BT == '101'

Figure A-59: AArch64_dbgbvr1_el1 bit assignments

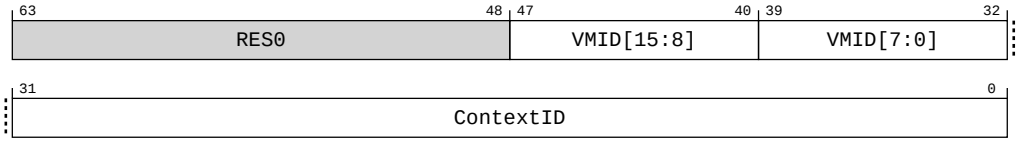


Table A-167: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR1_EL1.BT == '110'

Figure A-60: AArch64_dbgbvr1_el1 bit assignments

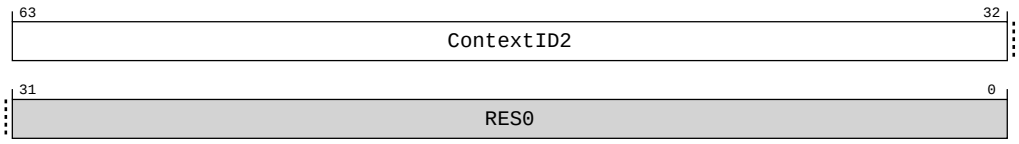


Table A-168: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR1_EL1.BT == '111'

Figure A-61: AArch64_dbgbvr1_el1 bit assignments

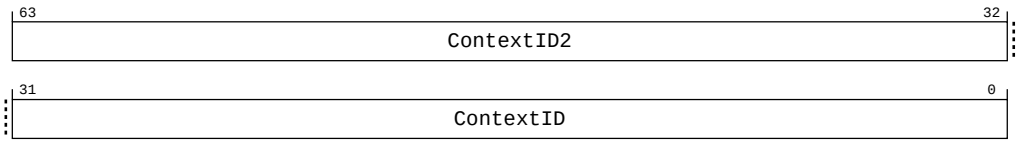


Table A-169: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

MRS <Xt>, DBGBVR1_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b100

MSR DBGBVR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b100

Accessibility

MRS <Xt>, DBGBVR1_EL1

```
if 1 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[1];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[1];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[1];
```

MSR DBGBVR1_EL1, <Xt>

```
if 1 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[1] = X[t, 64];
```

A.2.6 DBGBCR1_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register AArch64-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented, accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-62: AArch64_dbgbcr1_el1 bit assignments

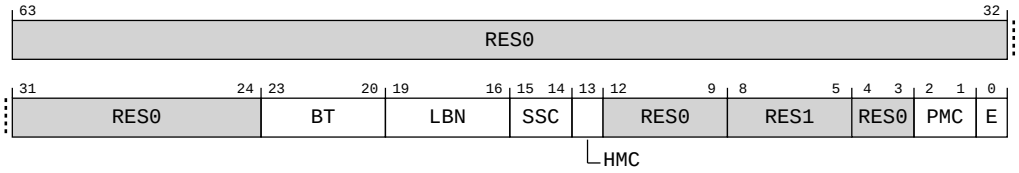


Table A-172: DBGBCR1_EL1 bit descriptions

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

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000, but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value **0b0011** As 0b0010, with linking enabled. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. AArch64-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGBCR<n>_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	xx
[0]	E	Enable breakpoint AArch64-DBGBVR<n>_EL1. **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

MRS <Xt>, DBGBCR1_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b101

MSR DBGBCR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b101

Accessibility

MRS <Xt>, DBGBCR1_EL1

```
if 1 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[1];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[1];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[1];
```

MSR DBGBCR1_EL1, <Xt>

```
if 1 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
```

```
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[1] = X[t, 64];
```

A.2.7 DBGWVR1_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register AArch64-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-63: AArch64_dbgwvr1_el1 bit assignments

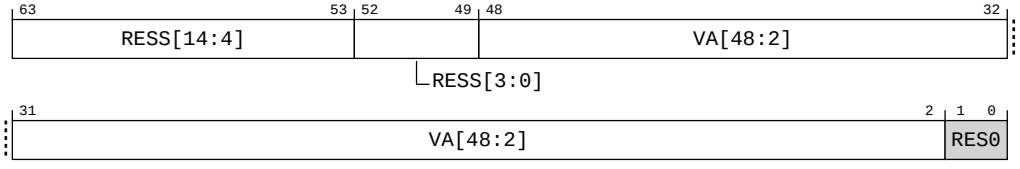


Table A-175: DBGWVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • It is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, DBGWVR1_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b110

MSR DBGWVR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b110

Accessibility

MRS <Xt>, DBGWVR1_EL1

```
if 1 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[1];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[1];
elsif PSTATE.EL == EL3 then
```

```
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[1];
```

MSR DBGWVR1_EL1, <Xt>

```
if 1 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[1] = X[t, 64];
```

A.2.8 DBGWCR1_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register AArch64-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-64: AArch64_dbgwcr1_el1 bit assignments

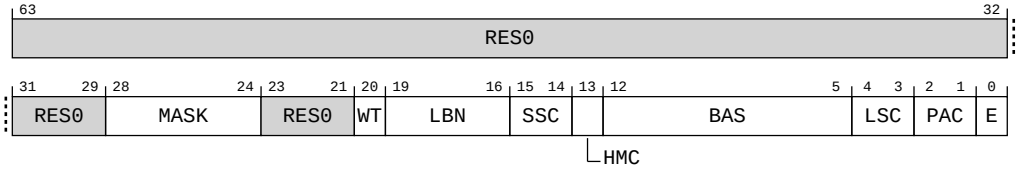


Table A-178: DBGWCR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	`5{x}`
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	`x`

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGWCR<n>_EL1.{SSC, HMC, PAC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by AArch64-DBGWVR<n>_EL1 is being watched. Table A-179: BAS description table 1 on page 288 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-180: BAS description table 2 on page 288 If AArch64-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>_EL1.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx

Bits	Name	Description	Reset
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table A-179: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table A-180: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

MRS <Xt>, DBGWCR1_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b111

MSR DBGWCR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0001	0b111

Accessibility

MRS <Xt>, DBGWCR1_EL1

```
if 1 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[1];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[1];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[1];
```

MSR DBGWCR1_EL1, <Xt>

```
if 1 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[1] = X[t, 64];
```

A.2.9 DBGBVR2_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register AArch64-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of AArch64-DBGBCR<n>_EL1.BT.

- When AArch64-DBGBCR<n>_EL1.BT is 0b000x, this register holds a virtual address.
- When AArch64-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of AArch64-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When AArch64-DBGBCR2_EL1.BT == '000x'

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

When AArch64-DBGBCR2_EL1.BT == '001x'

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

When AArch64-DBGBCR2_EL1.BT == '011x'

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

When AArch64-DBGBCR2_EL1.BT == '100x'

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

When AArch64-DBGBCR2_EL1.BT == '101x'

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

When AArch64-DBGBCR2_EL1.BT == '110x'

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

When AArch64-DBGBCR2_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-DBGBCR2_EL1.BT == '000'

Figure A-65: AArch64_dbgbvr2_el1 bit assignments

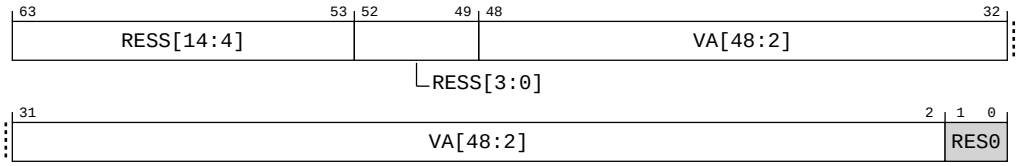


Table A-183: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • If the breakpoint is not context-aware, it is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR2_EL1.BT == '001'

Figure A-66: AArch64_dbgbvr2_el1 bit assignments

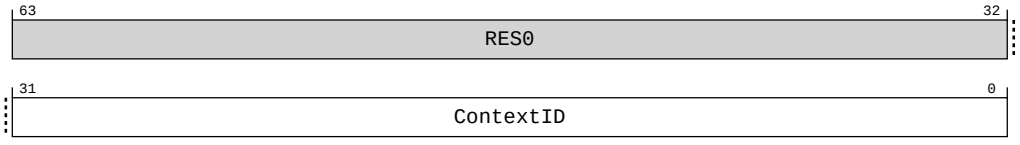


Table A-184: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR2_EL1.BT == '011'

Figure A-67: AArch64_dbgbvr2_el1 bit assignments

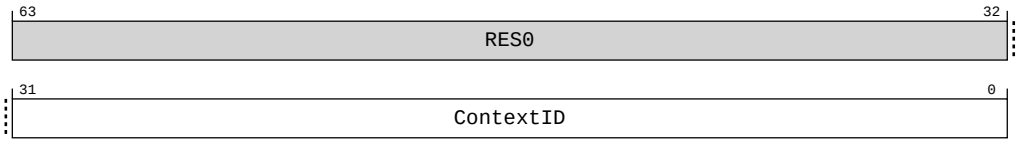


Table A-185: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR2_EL1.BT == '100'

Figure A-68: AArch64_dbgbvr2_el1 bit assignments

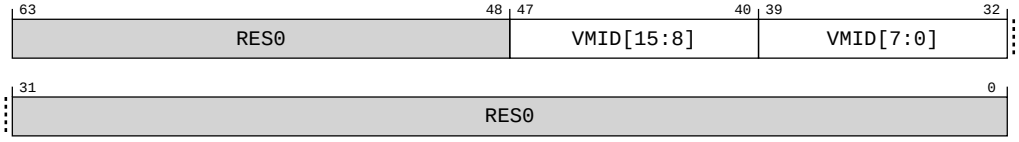


Table A-186: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}

Bits	Name	Description	Reset
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR2_EL1.BT == '101'

Figure A-69: AArch64_dbgbvr2_el1 bit assignments

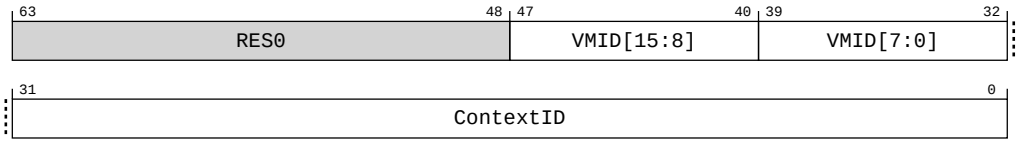


Table A-187: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR2_EL1.BT == '110'

Figure A-70: AArch64_dbgbvr2_el1 bit assignments

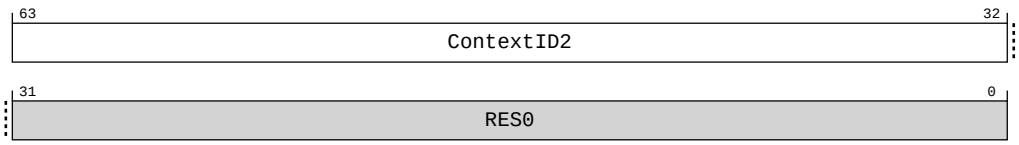


Table A-188: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR2_EL1.BT == '111'

Figure A-71: AArch64_dbgbvr2_el1 bit assignments

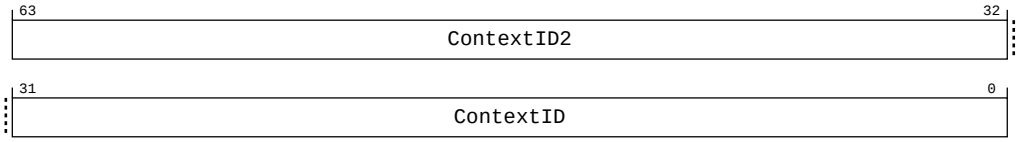


Table A-189: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

MRS <Xt>, DBGBVR2_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b100

MSR DBGBVR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b100

Accessibility

MRS <Xt>, DBGBVR2_EL1

```
if 2 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[2];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[2];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[2];
```

MSR DBGBVR2_EL1, <Xt>

```
if 2 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[2] = X[t, 64];
```

A.2.10 DBGBCR2_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register AArch64-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented, accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-72: AArch64_dbgbcr2_el1 bit assignments

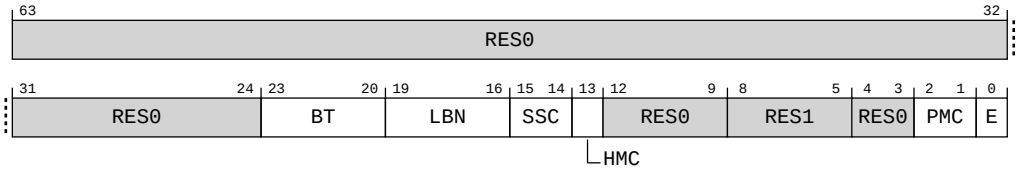


Table A-192: DBGBCR2_EL1 bit descriptions

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

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000, but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value **0b0011** As 0b0010, with linking enabled. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. AArch64-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGBCR<n>_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	xx
[0]	E	Enable breakpoint AArch64-DBGBVR<n>_EL1. **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

MRS <Xt>, DBGBCR2_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b101

MSR DBGBCR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b101

Accessibility

MRS <Xt>, DBGBCR2_EL1

```
if 2 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[2];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[2];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[2];
```

MSR DBGBCR2_EL1, <Xt>

```
if 2 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
```

```
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[2] = X[t, 64];
```

A.2.11 DBGWVR2_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register AArch64-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-73: AArch64_dbgwvr2_el1 bit assignments

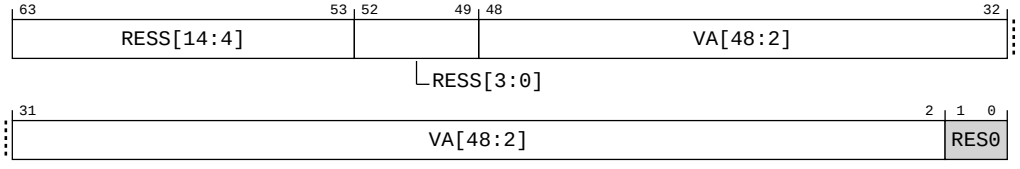


Table A-195: DBGWVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • It is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, DBGWVR2_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b110

MSR DBGWVR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b110

Accessibility

MRS <Xt>, DBGWVR2_EL1

```
if 2 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[2];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[2];
elsif PSTATE.EL == EL3 then
```

```
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[2];
```

MSR DBGWVR2_EL1, <Xt>

```
if 2 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[2] = X[t, 64];
```

A.2.12 DBGWCR2_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register AArch64-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-74: AArch64_dbgwcr2_el1 bit assignments

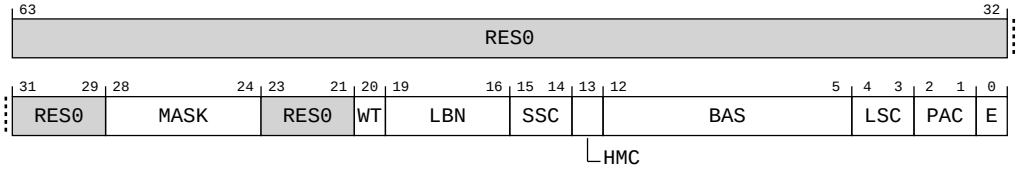


Table A-198: DBGWCR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	`5{x}`
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	`x`

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGWCR<n>_EL1.{SSC, HMC, PAC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by AArch64-DBGWVR<n>_EL1 is being watched. Table A-199: BAS description table 1 on page 305 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-200: BAS description table 2 on page 305 If AArch64-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>_EL1.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx

Bits	Name	Description	Reset
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table A-199: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table A-200: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

MRS <Xt>, DBGWCR2_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b111

MSR DBGWCR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b111

Accessibility

MRS <Xt>, DBGWCR2_EL1

```
if 2 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[2];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[2];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[2];
```

MSR DBGWCR2_EL1, <Xt>

```
if 2 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[2] = X[t, 64];
```

A.2.13 DBGBVR3_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register AArch64-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of AArch64-DBGBCR<n>_EL1.BT.

- When AArch64-DBGBCR<n>_EL1.BT is 0b000x, this register holds a virtual address.
- When AArch64-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of AArch64-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When AArch64-DBGBCR3_EL1.BT == '000x'

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

When AArch64-DBGBCR3_EL1.BT == '001x'

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

When AArch64-DBGBCR3_EL1.BT == '011x'

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

When AArch64-DBGBCR3_EL1.BT == '100x'

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

When AArch64-DBGBCR3_EL1.BT == '101x'

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

When AArch64-DBGBCR3_EL1.BT == '110x'

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

When AArch64-DBGBCR3_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-DBGBCR3_EL1.BT == '000'

Figure A-75: AArch64_dbgbvr3_el1 bit assignments

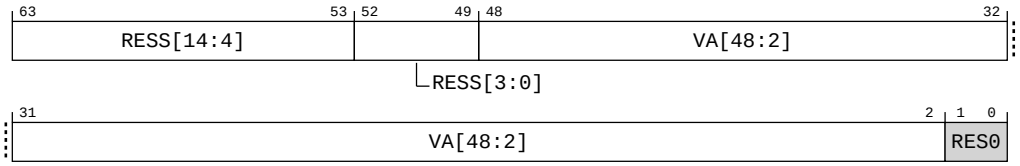


Table A-203: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • If the breakpoint is not context-aware, it is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR3_EL1.BT == '001'

Figure A-76: AArch64_dbgbvr3_el1 bit assignments

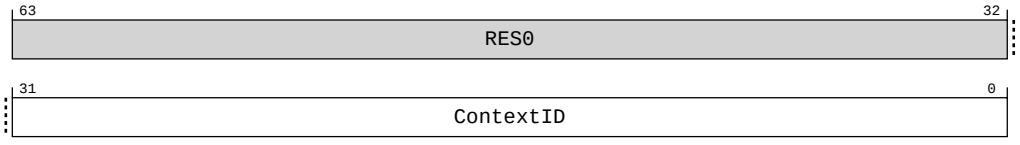


Table A-204: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR3_EL1.BT == '011'

Figure A-77: AArch64_dbgbvr3_el1 bit assignments

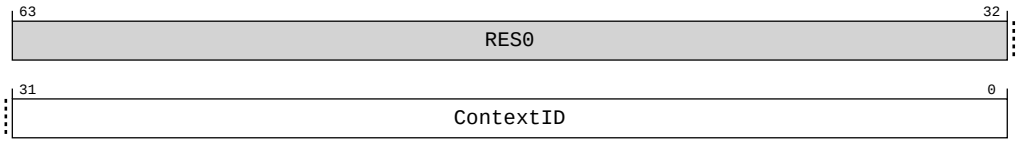


Table A-205: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR3_EL1.BT == '100'

Figure A-78: AArch64_dbgbvr3_el1 bit assignments

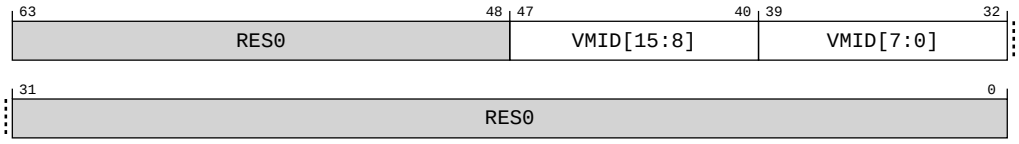


Table A-206: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}

Bits	Name	Description	Reset
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR3_EL1.BT == '101'

Figure A-79: AArch64_dbgbvr3_el1 bit assignments

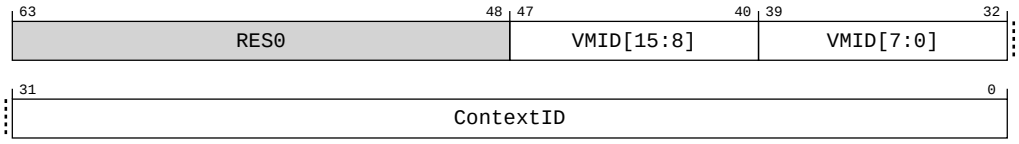


Table A-207: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR3_EL1.BT == '110'

Figure A-80: AArch64_dbgbvr3_el1 bit assignments

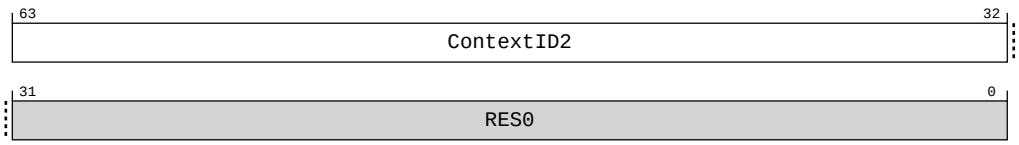


Table A-208: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR3_EL1.BT == '111'

Figure A-81: AArch64_dbgbvr3_el1 bit assignments

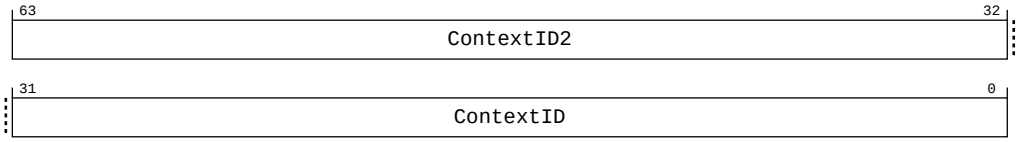


Table A-209: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

MRS <Xt>, DBGBVR3_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b100

MSR DBGBVR3_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b100

Accessibility

MRS <Xt>, DBGBVR3_EL1

```
if 3 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[3];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[3];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[3];
```

MSR DBGBVR3_EL1, <Xt>

```
if 3 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[3] = X[t, 64];
```

A.2.14 DBGBCR3_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register AArch64-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented, accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-82: AArch64_dbgbcr3_el1 bit assignments

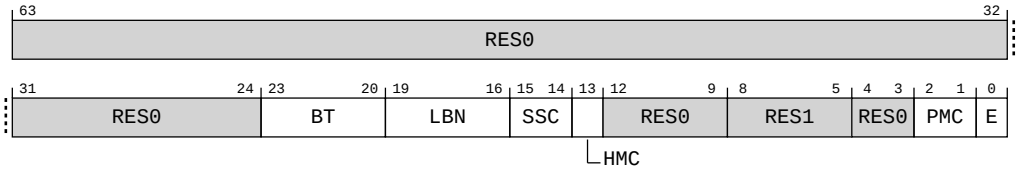


Table A-212: DBGBCR3_EL1 bit descriptions

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

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000, but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value **0b0011** As 0b0010, with linking enabled. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. AArch64-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGBCR<n>_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	xx
[0]	E	Enable breakpoint AArch64-DBGBVR<n>_EL1. **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

MRS <Xt>, DBGBCR3_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b101

MSR DBGBCR3_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b101

Accessibility

MRS <Xt>, DBGBCR3_EL1

```
if 3 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[3];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[3];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[3];
```

MSR DBGBCR3_EL1, <Xt>

```
if 3 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
```

```
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[3] = X[t, 64];
```

A.2.15 DBGWVR3_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register AArch64-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-83: AArch64_dbgwvr3_el1 bit assignments

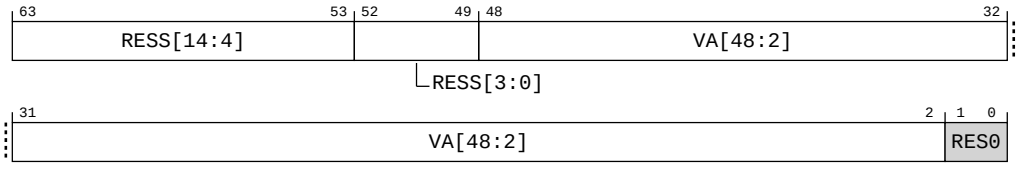


Table A-215: DBGWVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • It is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, DBGWVR3_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b110

MSR DBGWVR3_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b110

Accessibility

MRS <Xt>, DBGWVR3_EL1

```
if 3 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[3];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[3];
elsif PSTATE.EL == EL3 then
```

```
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[3];
```

MSR DBGWVR3_EL1, <Xt>

```
if 3 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[3] = X[t, 64];
```

A.2.16 DBGWCR3_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register AArch64-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-84: AArch64_dbgwcr3_el1 bit assignments

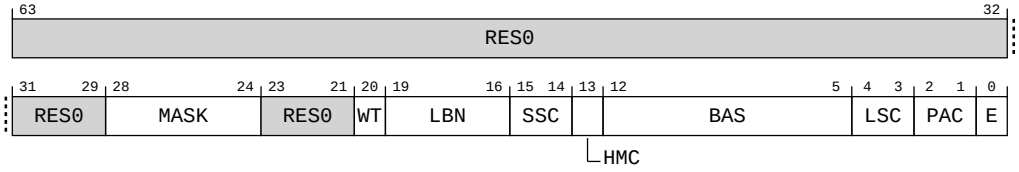


Table A-218: DBGWCR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	`5{x}`
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	`x`

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGWCR<n>_EL1.{SSC, HMC, PAC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by AArch64-DBGWVR<n>_EL1 is being watched. Table A-219: BAS description table 1 on page 322 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-220: BAS description table 2 on page 322 If AArch64-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting AArch64-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>_EL1.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx

Bits	Name	Description	Reset
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table A-219: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table A-220: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

MRS <Xt>, DBGWCR3_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b111

MSR DBGWCR3_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b111

Accessibility

MRS <Xt>, DBGWCR3_EL1

```
if 3 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[3];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[3];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[3];
```

MSR DBGWCR3_EL1, <Xt>

```
if 3 >= NUM_WATCHPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[3] = X[t, 64];
```

A.2.17 DBGBVR4_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register AArch64-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of AArch64-DBGBCR<n>_EL1.BT.

- When AArch64-DBGBCR<n>_EL1.BT is 0b000x, this register holds a virtual address.
- When AArch64-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of AArch64-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When AArch64-DBGBCR4_EL1.BT == '000x'

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

When AArch64-DBGBCR4_EL1.BT == '001x'

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

When AArch64-DBGBCR4_EL1.BT == '011x'

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

When AArch64-DBGBCR4_EL1.BT == '100x'

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

When AArch64-DBGBCR4_EL1.BT == '101x'

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

When AArch64-DBGBCR4_EL1.BT == '110x'

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

When AArch64-DBGBCR4_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-DBGBCR4_EL1.BT == '000'

Figure A-85: AArch64_dbgbvr4_el1 bit assignments

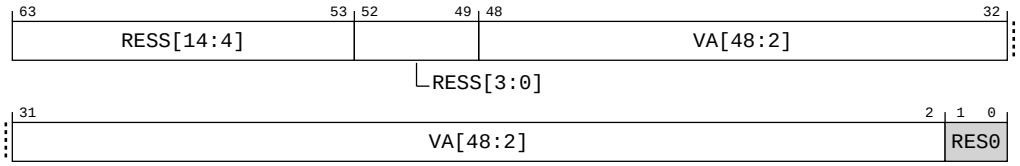


Table A-223: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • If the breakpoint is not context-aware, it is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR4_EL1.BT == '001'

Figure A-86: AArch64_dbgbvr4_el1 bit assignments

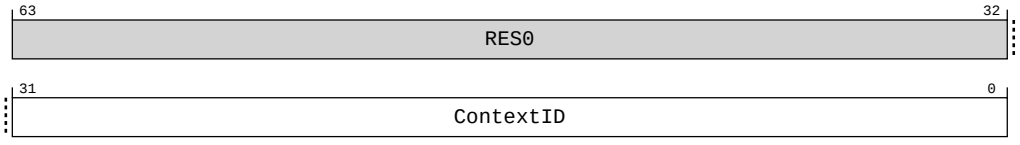


Table A-224: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR4_EL1.BT == '011'

Figure A-87: AArch64_dbgbvr4_el1 bit assignments

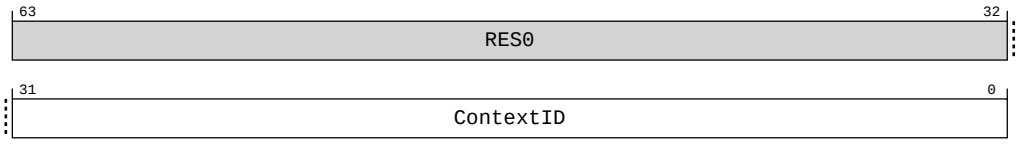


Table A-225: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR4_EL1.BT == '100'

Figure A-88: AArch64_dbgbvr4_el1 bit assignments

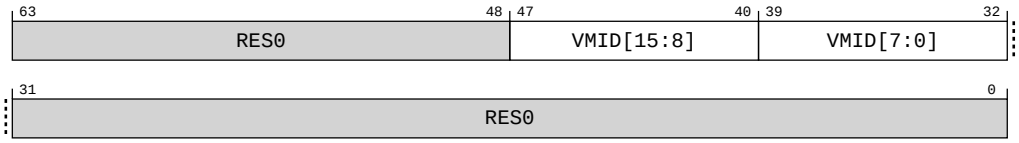


Table A-226: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}

Bits	Name	Description	Reset
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR4_EL1.BT == '101'

Figure A-89: AArch64_dbgbvr4_el1 bit assignments

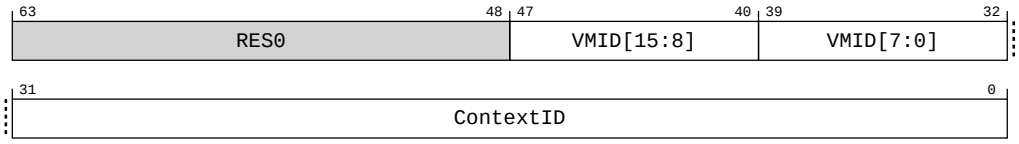


Table A-227: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR4_EL1.BT == '110'

Figure A-90: AArch64_dbgbvr4_el1 bit assignments

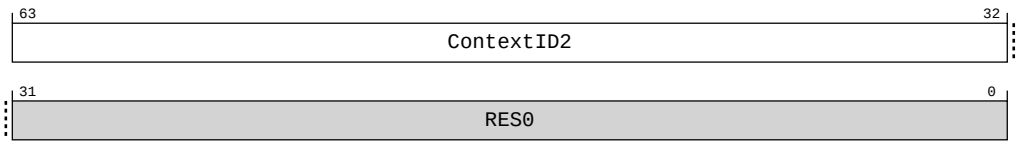


Table A-228: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR4_EL1.BT == '111'

Figure A-91: AArch64_dbgbvr4_el1 bit assignments

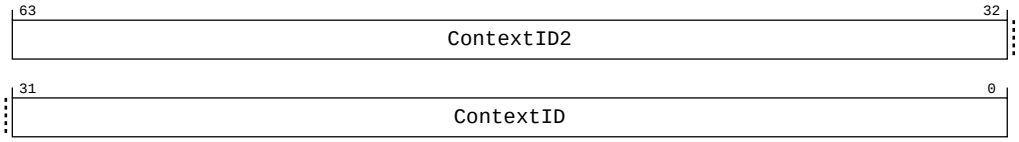


Table A-229: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

MRS <Xt>, DBGBVR4_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	0b100

MSR DBGBVR4_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	0b100

Accessibility

MRS <Xt>, DBGBVR4_EL1

```
if 4 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[4];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[4];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[4];
```

MSR DBGBVR4_EL1, <Xt>

```
if 4 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[4] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[4] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[4] = X[t, 64];
```

A.2.18 DBGBCR4_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register AArch64-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented, accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-92: AArch64_dbgbcr4_el1 bit assignments

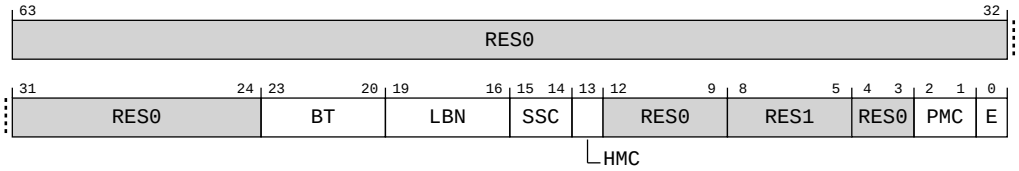


Table A-232: DBGBCR4_EL1 bit descriptions

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

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000, but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value **0b0011** As 0b0010, with linking enabled. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. AArch64-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGBCR<n>_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	xx
[0]	E	Enable breakpoint AArch64-DBGBVR<n>_EL1. **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

MRS <Xt>, DBGBCR4_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	0b101

MSR DBGBCR4_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	0b101

Accessibility

MRS <Xt>, DBGBCR4_EL1

```
if 4 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[4];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[4];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[4];
```

MSR DBGBCR4_EL1, <Xt>

```
if 4 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[4] = X[t, 64];
elsif PSTATE.EL == EL2 then
```

```
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[4] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[4] = X[t, 64];
```

A.2.19 DBGBVR5_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register AArch64-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of AArch64-DBGBCR<n>_EL1.BT.

- When AArch64-DBGBCR<n>_EL1.BT is 0b000x, this register holds a virtual address.
- When AArch64-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When AArch64-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of AArch64-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When AArch64-DBGBCR5_EL1.BT == '000x'

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

When AArch64-DBGBCR5_EL1.BT == '001x'

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

When AArch64-DBGBCR5_EL1.BT == '011x'

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

When AArch64-DBGBCR5_EL1.BT == '100x'

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

When AArch64-DBGBCR5_EL1.BT == '101x'

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

When AArch64-DBGBCR5_EL1.BT == '110x'

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

When AArch64-DBGBCR5_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When AArch64-DBGBCR5_EL1.BT == '000'

Figure A-93: AArch64_dbgbvr5_el1 bit assignments

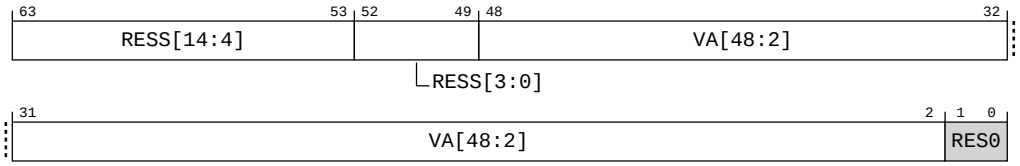


Table A-235: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply: • It is **CONSTRAINED UNPREDICTABLE** whether the PE ignores this field when comparing an address. • If the breakpoint is not context-aware, it is IMPLEMENTATION DEFINED whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR5_EL1.BT == '001'

Figure A-94: AArch64_dbgbvr5_el1 bit assignments

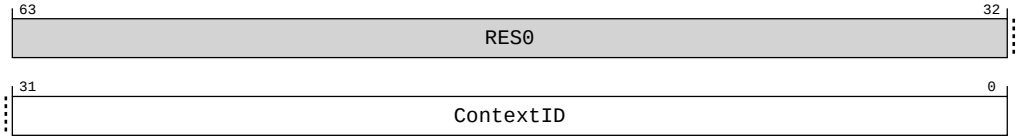


Table A-236: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR5_EL1.BT == '011'

Figure A-95: AArch64_dbgbvr5_el1 bit assignments

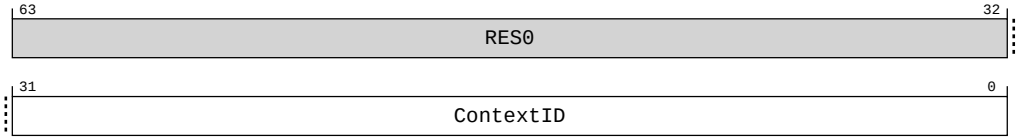


Table A-237: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR5_EL1.BT == '100'

Figure A-96: AArch64_dbgbvr5_el1 bit assignments

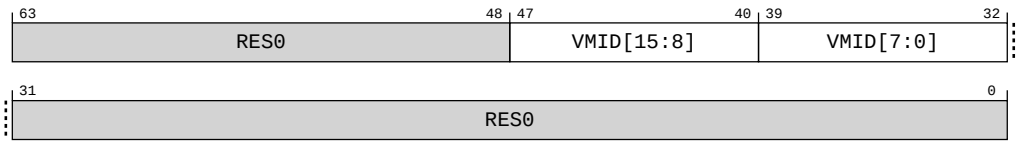


Table A-238: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR5_EL1.BT == '101'

Figure A-97: AArch64_dbgbvr5_el1 bit assignments

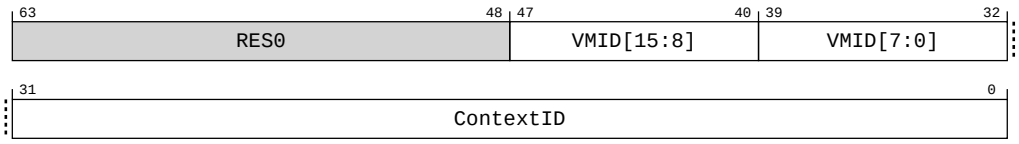


Table A-239: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When AArch64-DBGBCR5_EL1.BT == '110'

Figure A-98: AArch64_dbgbvr5_el1 bit assignments

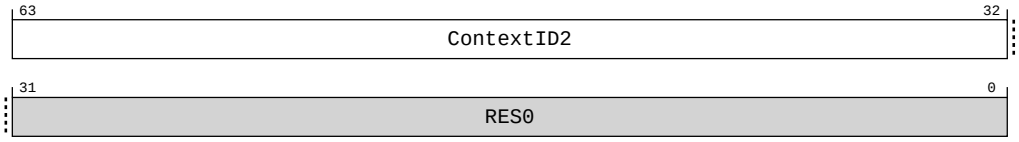


Table A-240: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When AArch64-DBGBCR5_EL1.BT == '111'

Figure A-99: AArch64_dbgbvr5_el1 bit assignments

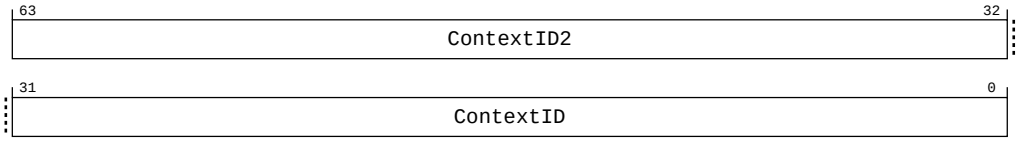


Table A-241: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

MRS <Xt>, DBGBVR5_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0101	0b100

MSR DBGBVR5_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0101	0b100

Accessibility

MRS <Xt>, DBGBVR5_EL1

```
if 5 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
```

```
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[5];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[5];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBVR_EL1[5];
```

MSR DBGBVR5_EL1, <Xt>

```
if 5 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[5] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[5] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBVR_EL1[5] = X[t, 64];
```

A.2.20 DBGBCR5_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register AArch64-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented, accesses to this register are UNDEFINED.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-100: AArch64_dbgbcr5_el1 bit assignments

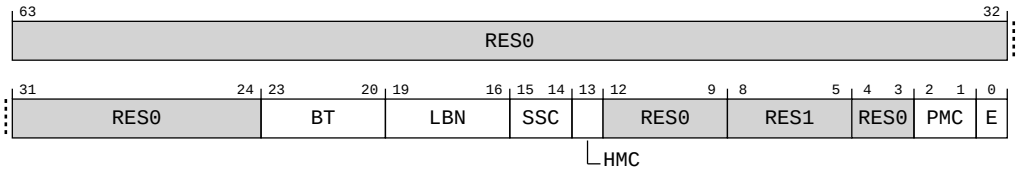


Table A-244: DBGBCR5_EL1 bit descriptions

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

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000, but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value **0b0011** As 0b0010, with linking enabled. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. AArch64-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved value, see *Reserved DBGBCR<n>_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained. For more information on the operation of these fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information, see DBGBCR<n>_EL1.SSC.	xx
[0]	E	Enable breakpoint AArch64-DBGBVR<n>_EL1. **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

MRS <Xt>, DBGBCR5_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0101	0b101

MSR DBGBCR5_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0101	0b101

Accessibility

MRS <Xt>, DBGBCR5_EL1

```
if 5 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[5];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[5];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[5];
```

MSR DBGBCR5_EL1, <Xt>

```
if 5 >= NUM_BREAKPOINTS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[5] = X[t, 64];
elsif PSTATE.EL == EL2 then
```

```
    if MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[5] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[5] = X[t, 64];
```

A.2.21 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 registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-101: AArch64_imp_idata0_el3 bit assignments

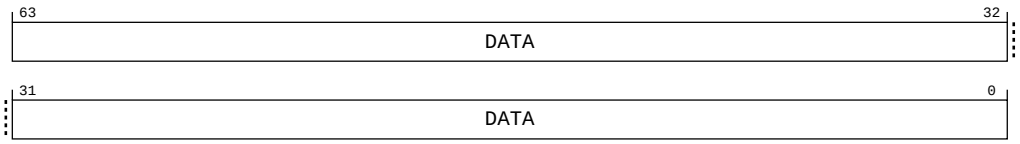


Table A-247: IMP_IDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	64{x}

Access

MRS <Xt>, S3_6_C15_C0_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0000	0b000

Accessibility

MRS <Xt>, S3_6_C15_C0_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_IDATA0_EL3;
```

A.2.22 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 registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-102: AArch64_imp_idata1_el3 bit assignments

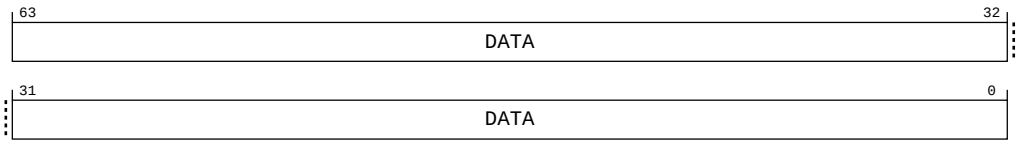


Table A-249: IMP_IDATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	64{x}

Access

MRS <Xt>, S3_6_C15_C0_1

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

Accessibility

MRS <Xt>, S3_6_C15_C0_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_IDATA1_EL3;
```

A.2.23 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 registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-103: AArch64_imp_idata2_el3 bit assignments

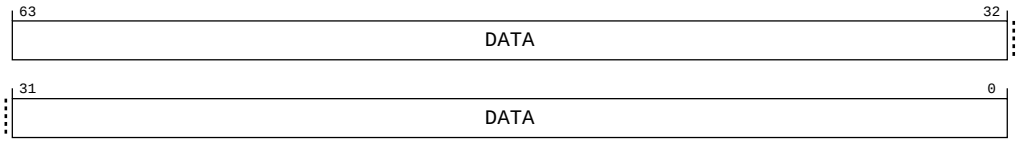


Table A-251: IMP_IDATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	64{x}

Access

MRS <Xt>, S3_6_C15_C0_2

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

Accessibility

MRS <Xt>, S3_6_C15_C0_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_IDATA2_EL3;
```

A.2.24 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 registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-104: AArch64_imp_ddata0_el3 bit assignments

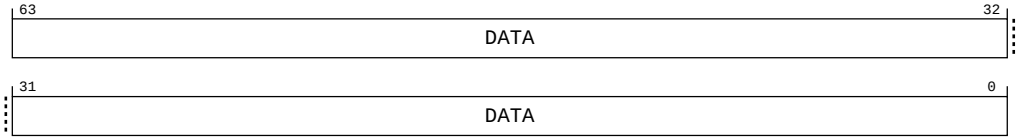


Table A-253: IMP_DDATA0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	64{x}

Access

MRS <Xt>, S3_6_C15_C1_0

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

Accessibility

MRS <Xt>, S3_6_C15_C1_0

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

```
    X[t, 64] = IMP_DDATA0_EL3;
```

A.2.25 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 registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-105: AArch64_imp_ddata1_el3 bit assignments

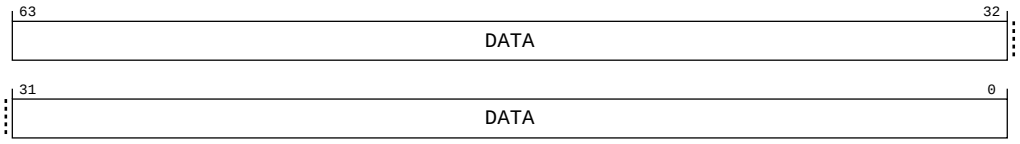


Table A-255: IMP_DDATA1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	64{x}

Access

MRS <Xt>, S3_6_C15_C1_1

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

Accessibility

MRS <Xt>, S3_6_C15_C1_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_DDATA1_EL3;
```

A.2.26 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 registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-106: AArch64_imp_ddata2_el3 bit assignments

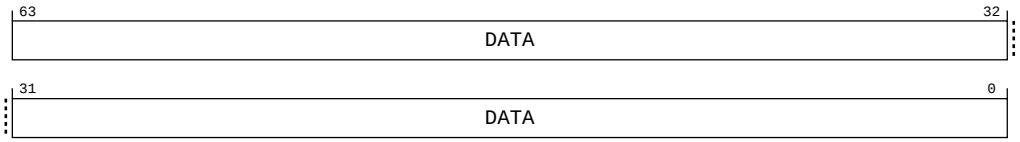


Table A-257: IMP_DDATA2_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	64{x}

Access

MRS <Xt>, S3_6_C15_C1_2

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

Accessibility

MRS <Xt>, S3_6_C15_C1_2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_DDATA2_EL3;
```

A.3 AArch64 System instructions summary

The summary table provides an overview of all System instructions in the core. For more information about a register, click the register name in the table.

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

Table A-259: System instructions summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SYS_IMP_RAMINDEX	1	6	C15	C0	0	–	64-bit	RAM Index

A.3.1 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 instructions

Access type

See bit descriptions

Bit descriptions

Figure A-107: AArch64_sys_imp_ramindex bit assignments

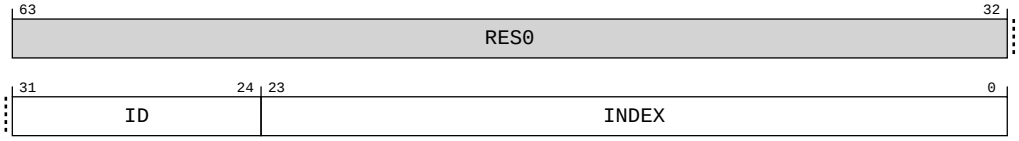


Table A-260: SYS_IMP_RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	ID	RAM ID (See Chapter 10)	8{x}
[23:0]	INDEX	RAM Index (See Chapter 10)	24{x}

Access

SYS #6, C15, C0, #0{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1111	0b0000	0b000

Accessibility

SYS #6, C15, C0, #0{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_RAMINDEX(X[t, 64]);
```

A.4 AArch64 Identification registers summary

The summary table provides an overview of all Identification registers in the core. For more information about a register, click the register name in the table.

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

Table A-262: Identification registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	0	C0	C0	0	–	64-bit	Main ID Register
MPIDR_EL1	3	0	C0	C0	5	–	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	0	C0	C0	6	–	64-bit	Revision ID Register
ID_PFR0_EL1	3	0	C0	C1	0	–	64-bit	AArch32 Processor Feature Register 0
ID_PFR1_EL1	3	0	C0	C1	1	–	64-bit	AArch32 Processor Feature Register 1
ID_DFR0_EL1	3	0	C0	C1	2	–	64-bit	AArch32 Debug Feature Register 0
ID_AFR0_EL1	3	0	C0	C1	3	–	64-bit	AArch32 Auxiliary Feature Register 0
ID_MMFR0_EL1	3	0	C0	C1	4	–	64-bit	AArch32 Memory Model Feature Register 0
ID_MMFR1_EL1	3	0	C0	C1	5	–	64-bit	AArch32 Memory Model Feature Register 1
ID_MMFR2_EL1	3	0	C0	C1	6	–	64-bit	AArch32 Memory Model Feature Register 2
ID_MMFR3_EL1	3	0	C0	C1	7	–	64-bit	AArch32 Memory Model Feature Register 3
ID_ISAR0_EL1	3	0	C0	C2	0	–	64-bit	AArch32 Instruction Set Attribute Register 0
ID_ISAR1_EL1	3	0	C0	C2	1	–	64-bit	AArch32 Instruction Set Attribute Register 1
ID_ISAR2_EL1	3	0	C0	C2	2	–	64-bit	AArch32 Instruction Set Attribute Register 2
ID_ISAR3_EL1	3	0	C0	C2	3	–	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	0	C0	C2	4	–	64-bit	AArch32 Instruction Set Attribute Register 4
ID_ISAR5_EL1	3	0	C0	C2	5	–	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	0	C0	C2	6	–	64-bit	AArch32 Memory Model Feature Register 4
ID_ISAR6_EL1	3	0	C0	C2	7	–	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	0	C0	C3	0	–	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	0	C0	C3	1	–	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	0	C0	C3	2	–	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	0	C0	C3	4	–	64-bit	AArch32 Processor Feature Register 2
ID_DFR1_EL1	3	0	C0	C3	5	–	64-bit	Debug Feature Register 1
ID_MMFR5_EL1	3	0	C0	C3	6	–	64-bit	AArch32 Memory Model Feature Register 5
ID_AA64PFR0_EL1	3	0	C0	C4	0	–	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	0	C0	C4	1	–	64-bit	AArch64 Processor Feature Register 1
ID_AA64PFR2_EL1	3	0	C0	C4	2	–	64-bit	AArch64 Processor Feature Register 2
ID_AA64ZFR0_EL1	3	0	C0	C4	4	–	64-bit	SVE Feature ID register 0
ID_AA64DFR0_EL1	3	0	C0	C5	0	–	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	0	C0	C5	1	–	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	0	C0	C5	4	–	64-bit	AArch64 Auxiliary Feature Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ID_AA64AFR1_EL1	3	0	C0	C5	5	–	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	0	C0	C6	0	–	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	0	C0	C6	1	–	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64ISAR2_EL1	3	0	C0	C6	2	–	64-bit	AArch64 Instruction Set Attribute Register 2
ID_AA64MMFR0_EL1	3	0	C0	C7	0	–	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	0	C0	C7	1	–	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	0	C0	C7	2	–	64-bit	AArch64 Memory Model Feature Register 2
MPAMIDR_EL1	3	0	C10	C4	4	–	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	0	C15	C0	0	–	64-bit	CPU Configuration Register
CCSIDR_EL1	3	1	C0	C0	0	–	64-bit	Current Cache Size ID Register
CLIDR_EL1	3	1	C0	C0	1	–	64-bit	Cache Level ID Register
CCSIDR2_EL1	3	1	C0	C0	2	–	64-bit	Current Cache Size ID Register 2
GMID_EL1	3	1	C0	C0	4	–	64-bit	Multiple tag transfer ID register
CSSELR_EL1	3	2	C0	C0	0	–	64-bit	Cache Size Selection Register
CTR_EL0	3	3	C0	C0	1	–	64-bit	Cache Type Register
DCZID_EL0	3	3	C0	C0	7	–	64-bit	Data Cache Zero ID register
VPIDR_EL2	3	4	C0	C0	0	–	64-bit	Virtualization Processor ID Register
VMPIDR_EL2	3	4	C0	C0	5	–	64-bit	Virtualization Multiprocessor ID Register

A.4.1 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0100 0001 0001 1111 1101 0100 1110 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-108: AArch64_midr_el1 bit assignments

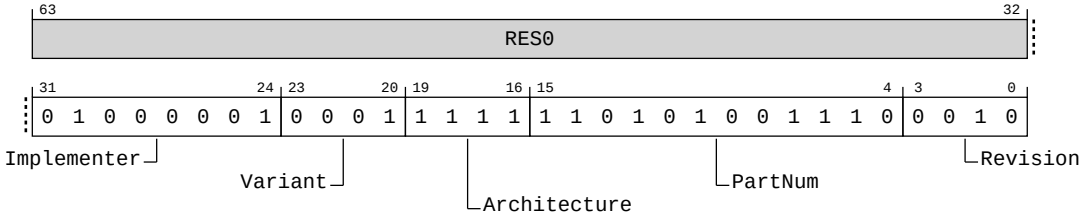


Table A-263: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	Implementer	Indicates the implementer code. This value is: **0b01000001** Arm Limited.	0x41
[23:20]	Variant	Variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product. **0b0001** r1p2	0b0001
[19:16]	Architecture	Indicates the architecture code. This value is: **0b1111** Architecture is defined by ID registers	0b1111
[15:4]	PartNum	Primary Part Number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. **0b110101001110** Cortex-X3	0xD4E
[3:0]	Revision	Revision number for the device. **0b0010** r1p2	0b0010

Access

MRS <Xt>, MIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b000

Accessibility

MRS <Xt>, MIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() then
        X[t, 64] = VPIDR_EL2;
    else
        X[t, 64] = MIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MIDR_EL1;
```

A.4.2 MPIDR_EL1, Multiprocessor Affinity Register

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

Configurations

In a uniprocessor system, Arm recommends that each Aff<n> field of this register returns a value of 0.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx xxx1 xxxx xxxx xxxx xxxx 0000 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-109: AArch64_mpidr_el1 bit assignments

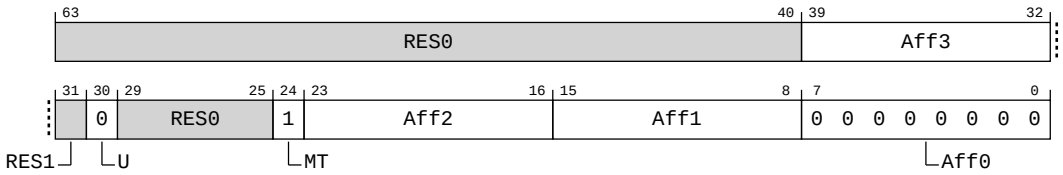


Table A-265: MPIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:32]	Aff3	Affinity level 3. See the description of Aff0 for more information. The value will be determined by the CLUSTERIDAFF3 configuration pins.	`8{x}`
[31]	**RES1**	Reserved	**RES1**
[30]	U	Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. **0b0** Processor is part of a multiprocessor system.	`0b0`
[29:25]	**RES0**	Reserved	**RES0**
[24]	MT	Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of Aff0 for more information about affinity levels. **0b1** Performance of PEs with different affinity level 0 values, and the same values for affinity level 1 and higher, is very interdependent.	`0b1`
[23:16]	Aff2	Affinity level 2. See the description of Aff0 for more information. The value will be determined by the CLUSTERIDAFF2 configuration pins.	`8{x}`
[15:8]	Aff1	Affinity level 1. See the description of Aff0 for more information. Value read from the CPUID configuration pins. Identification number for each CPU in an cluster counting from zero.	`8{x}`
[7:0]	Aff0	Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or AArch64-MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole. **0b00000000** Only one thread.	`0x00`

Access

MRS <Xt>, MPIDR_EL1

op0	op1	CRn	CRm	op2
`0b11`	`0b000`	`0b0000`	`0b0000`	`0b101`

Accessibility

MRS <Xt>, MPIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() then
        X[t, 64] = VMPIDR_EL2;
    else
        X[t, 64] = MPIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MPIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPIDR_EL1;
```

A.4.3 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 Cortex®-X3 core. Refer to the Cortex®-X3 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 registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-110: AArch64_revidr_el1 bit assignments

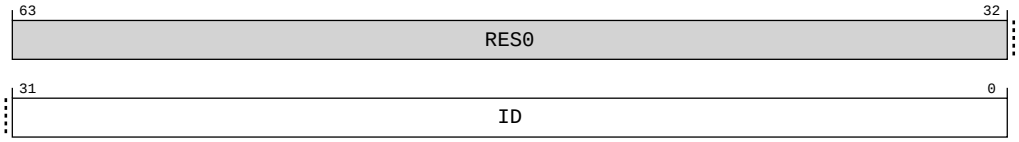


Table A-267: REVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ID	None	32{x}

Access

MRS <Xt>, REVIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b110

Accessibility

MRS <Xt>, REVIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = REVIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = REVIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = REVIDR_EL1;
```

A.4.4 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0

Provides additional information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

The external register ext-EDPFR gives information from this register.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0001 0010 xxxx 0001 0001 0001 0001 0001 0010 xxxx 0001 0001 0001 0001 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-111: AArch64_id_aa64pfr0_el1 bit assignments

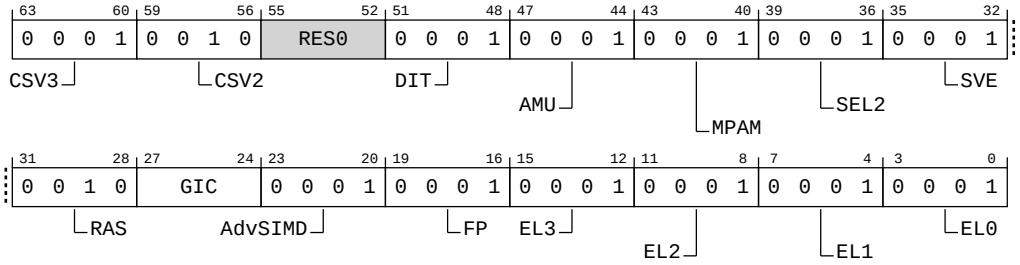


Table A-269: ID_AA64PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	CSV3	Speculative use of faulting data. Defined values are: **0b0001** Data loaded under speculation with a permission or domain fault cannot be used to form an address, generate condition codes, or generate SVE predicate values to be used by other instructions in the speculative sequence. The execution timing of any other instructions in the speculative sequence is not a function of the data loaded under speculation.	0b0001
[59:56]	CSV2	Speculative use of out of context branch targets. Defined values are: **0b0010** FEAT_CSV2_2 is implemented, but FEAT_CSV2_3 is not implemented.	0b0010
[55:52]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[51:48]	DIT	Data Independent Timing. Defined values are: **0b0001** AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.	0b0001
[47:44]	AMU	Indicates support for Activity Monitors Extension. Defined values are: **0b0001** FEAT_AMUv1 is implemented.	0b0001
[43:40]	MPAM	Indicates the major version number of support for the MPAM Extension. Defined values are: **0b0001** The major version number of the MPAM extension is 1.	0b0001
[39:36]	SEL2	Secure EL2. Defined values are: **0b0001** Secure EL2 is implemented.	0b0001
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0b0001** SVE architectural state and programmers' model are implemented.	0b0001
[31:28]	RAS	RAS Extension version. Defined values are: **0b0010** 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.	0b0010
[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.	The reset values can be the following: 0b0000, 0b0011, respective to the value.
[23:20]	AdvSIMD	Advanced SIMD. Defined values are: **0b0001** Advanced SIMD is implemented, including support for half-precision floating-point arithmetic.	0b0001
[19:16]	FP	Floating-point. Defined values are: **0b0001** Floating-point, including support for half-precision floating-point arithmetic, is implemented.	0b0001
[15:12]	EL3	EL3 Exception level handling. Defined values are: **0b0001** EL3 can be executed in AArch64 state only.	0b0001
[11:8]	EL2	EL2 Exception level handling. Defined values are: **0b0001** EL2 can be executed in AArch64 state only.	0b0001

Bits	Name	Description	Reset
[7:4]	EL1	EL1 Exception level handling. Defined values are: **0b0001** EL1 can be executed in AArch64 state only.	0b0001
[3:0]	EL0	EL0 Exception level handling. Defined values are: **0b0001** EL0 can be executed in AArch64 state only.	0b0001

Access

MRS <Xt>, ID_AA64PFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b000

Accessibility

MRS <Xt>, ID_AA64PFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64PFR0_EL1;
```

A.4.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 xxxx xxxx xxxx 0000 xxxx 0010 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-112: AArch64_id_aa64pfr1_el1 bit assignments

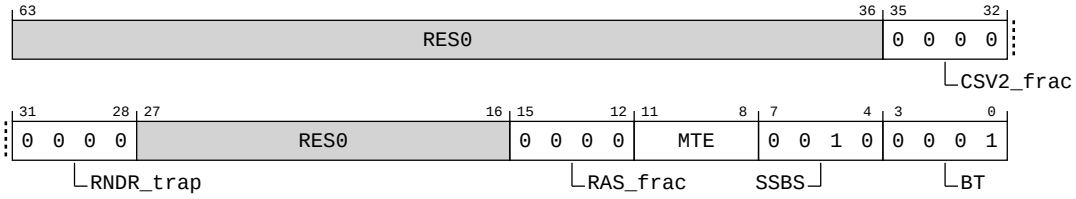


Table A-271: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:36]	**RES0**	Reserved	**RES0**
[35:32]	CSV2_frac	CSV2 fractional field. Defined values are: **0b0000** Either AArch64-ID_AA64PFR0_EL1.CSV2 is not 0b0001, or the implementation does not disclose whether FEAT_CSV2_1p1 is implemented. FEAT_CSV2_1p2 is not implemented.	0b0000
[31:28]	RNDR_trap	Random Number trap to EL3 field. Defined values are: **0b0000** Trapping of AArch64-RNDR and AArch64-RNDRRS to EL3 is not supported.	0b0000
[27:16]	**RES0**	Reserved	**RES0**
[15:12]	RAS_frac	RAS Extension fractional field. Defined values are: **0b0000** If AArch64-ID_AA64PFR0_EL1.RAS == 0b0001, RAS Extension implemented.	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. **0b0011** Memory Tagging Extension is implemented with support for asymmetric Tag Check Fault handling. This value is reported when the BROADCASTMTE input is HIGH.	The reset values can be the following: 0b0001, 0b0011, respective to the value.
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. Defined values are: **0b0010** As 0b0001, and adds the MSR and MRS instructions to directly read and write the PSTATE.SSBS field.	0b0010
[3:0]	BT	Branch Target Identification mechanism support in AArch64 state. Defined values are: **0b0001** The Branch Target Identification mechanism is implemented.	0b0001

Access

MRS <Xt>, ID_AA64PFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b001

Accessibility

MRS <Xt>, ID_AA64PFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64PFR1_EL1;
```

A.4.6 ID_AA64PFR2_EL1, AArch64 Processor Feature Register 2

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-113: AArch64_id_aa64pfr2_el1 bit assignments

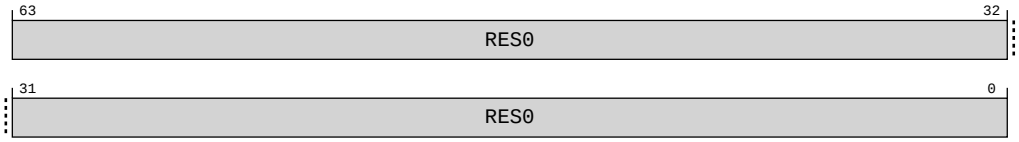


Table A-273: ID_AA64PFR2_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64PFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b010

Accessibility

MRS <Xt>, ID_AA64PFR2_EL1

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

```
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    X[t, 64] = ID_AA64PFR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64PFR2_EL1;
```

A.4.7 ID_AA64ZFR0_EL1, SVE Feature ID register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension, when FEAT_SVE is implemented.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx 0000 0000 xxxx 0001 xxxx xxxx xxxx xxxx xxxx 0001 0001 xxxx xxxx xxxx 0001

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-114: AArch64_id_aa64zfr0_el1 bit assignments

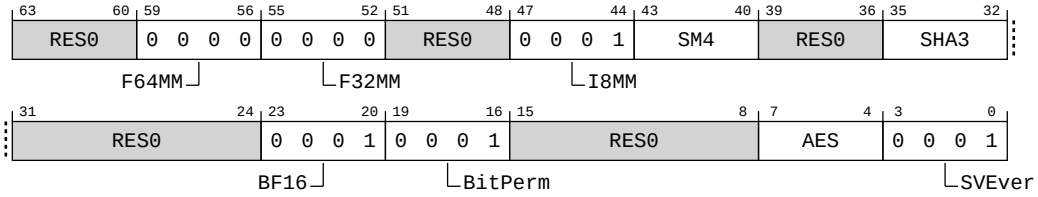


Table A-275: ID_AA64ZFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	**RES0**	Reserved	**RES0**
[59:56]	F64MM	Indicates support for SVE FP64 double-precision floating-point matrix multiplication instructions. Defined values are: **0b0000** Double-precision matrix multiplication and related SVE instructions are not implemented. All other values are reserved. FEAT_F64MM implements the functionality identified by 0b0001. From Armv8.2, the permitted values are 0b0000 and 0b0001.	0b0000
[55:52]	F32MM	Indicates support for the SVE FP32 single-precision floating-point matrix multiplication instruction. Defined values are: **0b0000** Single-precision matrix multiplication instruction is not implemented. All other values are reserved. FEAT_F32MM implements the functionality identified by 0b0001. From Arm v8.2, the permitted values are 0b0000 and 0b0001.	0b0000
[51:48]	**RES0**	Reserved	**RES0**
[47:44]	I8MM	Indicates support for SVE Int8 matrix multiplication instructions. Defined values are: **0b0001** SVE SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	0b0001
[43:40]	SM4	Indicates support for SVE SM4 instructions. Defined values are: **0b0000** SVE2 SM4 instructions are not implemented. This value is reported when the Cryptographic Extension is not implemented or is disabled, or SM3/SM4 Cryptographic extensions are not implemented or are disabled. **0b0001** SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when the Cryptographic Extension is implemented and SM3/SM4 Cryptographic instructions are enabled.	The reset values can be the following: 0b0000, 0b0001, respective to the value.

Bits	Name	Description	Reset
[39:36]	**RES0**	Reserved	**RES0**
[35:32]	SHA3	Indicates support for the SVE SHA3 instructions. Defined values are: **0b0000** SVE2 SHA-3 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SVE2 RAX1 instruction is implemented. This value is reported when Cryptographic extensions are implemented and enabled.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[31:24]	**RES0**	Reserved	**RES0**
[23:20]	BF16	Indicates support for SVE BFloat16 instructions. Defined values are: **0b0001** SVE BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	0b0001
[19:16]	BitPerm	Indicates support for SVE bit permute instructions. Defined values are: **0b0001** SVE BDEP, BEXT, and BGRP instructions are implemented.	0b0001
[15:8]	**RES0**	Reserved	**RES0**
[7:4]	AES	Indicates support for SVE AES instructions. Defined values are: **0b0000** SVE2-AES instructions are not implemented. This value is reported when 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.	The reset values can be the following: 0b0000, 0b0010, respective to the value.
[3:0]	SVEver	Indicates support for SVE instructions when FEAT_SVE is implemented. Defined values are: **0b0001** As 0b0000, and adds the mandatory SVE2 instructions.	0b0001

Access

MRS <Xt>, ID_AA64ZFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b100

Accessibility

MRS <Xt>, ID_AA64ZFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    else
        X[t, 64] = ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ZFR0_EL1;
```

A.4.8 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

The external register ext-EDDFR gives information from this register.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx 0001 0001 1111 0010 0001 xxxx 0011 xxxx 0101 0110 0001 1001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-115: AArch64_id_aa64dfr0_el1 bit assignments

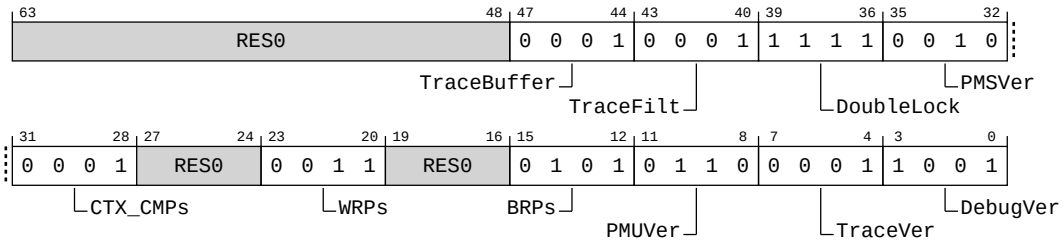


Table A-277: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:44]	TraceBuffer	Trace Buffer Extension. Defined values are: **0b0001** Trace Buffer Extension implemented.	0b0001
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: **0b0001** Armv8.4 Self-hosted Trace Extension implemented.	0b0001
[39:36]	DoubleLock	OS Double Lock implemented. Defined values are: **0b1111** OS Double Lock not implemented. AArch64-OSDLR_EL1 is **RAZ/WI**.	0b1111
[35:32]	PMSVer	Statistical Profiling Extension version. Defined values are: **0b0010** As 0b0001, and adds: • Support for the Event packet Alignment flag. • If FEAT_SVE is implemented, support for the Scalable Vector extensions to Statistical Profiling.	0b0010
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. **0b0001** Two context-aware breakpoints are included	0b0001
[27:24]	**RES0**	Reserved	**RES0**
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0b0000 is reserved. **0b0011** Four Watchpoints	0b0011
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0b0000 is reserved. **0b0101** Six Breakpoints	0b0101
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **0b0110** Performance Monitors Extension implemented, PMUv3 for Armv8.5	0b0110
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a trace unit is implemented. Defined values are: **0b0001** Trace unit System registers implemented.	0b0001
[3:0]	DebugVer	Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are: **0b1001** Armv8.4 debug architecture, FEAT_Debugv8p4.	0b1001

Access

MRS <Xt>, ID_AA64DFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b000

Accessibility

MRS <Xt>, ID_AA64DFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64DFR0_EL1;
```

A.4.9 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1

Provides top level information about the debug system in AArch64 state.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-116: AArch64_id_aa64dfr1_el1 bit assignments

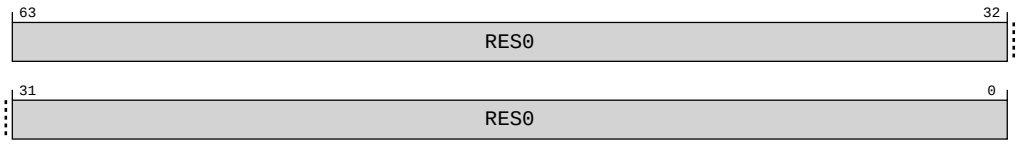


Table A-279: ID_AA64DFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64DFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b001

Accessibility

MRS <Xt>, ID_AA64DFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64DFR1_EL1;
```

A.4.10 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-117: AArch64_id_aa64afr0_el1 bit assignments

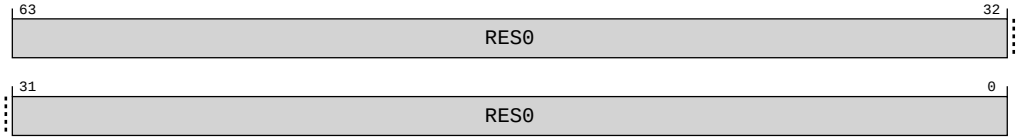


Table A-281: ID_AA64AFR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64AFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b100

Accessibility

MRS <Xt>, ID_AA64AFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    else
        X[t, 64] = ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64AFR0_EL1;
```

A.4.11 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

Reserved for future expansion of information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-118: AArch64_id_aa64afr1_el1 bit assignments

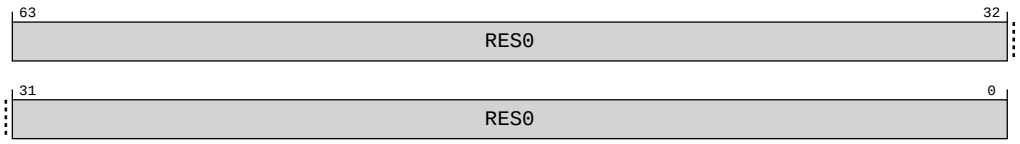


Table A-283: ID_AA64AFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64AFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b101

Accessibility

MRS <Xt>, ID_AA64AFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64AFR1_EL1;
```

A.4.12 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0000 0010 0010 0001 0001 xxxx xxxx xxxx 0001 0000 0010 0001 xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-119: AArch64_id_aa64isar0_el1 bit assignments

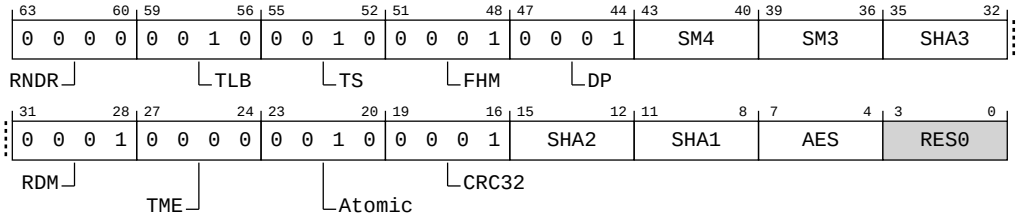


Table A-285: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RNDR	Indicates support for Random Number instructions in AArch64 state. When FEAT_RNG_TRAP is implemented, the value returned by a direct read of ID_AA64ISAR0_EL1.RNDR is further controlled by the value of AArch64-SCR_EL3.TRNDR. Defined values are: **0b0000** No Random Number instructions are implemented.	0b0000
[59:56]	TLB	Indicates support for Outer Shareable and TLB range maintenance instructions. Defined values are: **0b0010** Outer Shareable and TLB range maintenance instructions are implemented.	0b0010
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	0b0010
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: **0b0001** FMLAL and FMLSL instructions are implemented.	0b0001
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **0b0001** UDOT and SDOT instructions implemented.	0b0001

Bits	Name	Description	Reset
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **0b0000** When the Cryptographic Extension is not implemented or is disabled or the SM3/SM4 Cryptographic instructions are disabled, then SM4 instructions are not implemented. **0b0001** When the Cryptographic Extension is implemented and the SM3/SM4 Cryptographic instructions are enabled, then SM4 instructions SM4E and SM4EKEY are implemented.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: **0b0000** When the Cryptographic Extension is not implemented or is disabled or the SM3/SM4 Cryptographic instructions are disabled, then SM3 instructions are not implemented. **0b0001** When the Cryptographic Extension is implemented and the SM3/SM4 Cryptographic instructions are enabled, then SM3 instructions SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 are implemented.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: **0b0000** 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.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: **0b0001** SQRDMLAH and SQRDMLSH instructions implemented.	0b0001
[27:24]	TME	Indicates support for TME instructions. Defined values are: **0b0000** TME instructions are not implemented.	0b0000
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. Defined values are: **0b0010** LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	0b0010
[19:16]	CRC32	Indicates support for CRC32 instructions in AArch64 state. Defined values are: **0b0001** CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.	0b0001

Bits	Name	Description	Reset
[15:12]	SHA2	Indicates support for SHA2 instructions 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	The reset values can be the following: 0b0000, 0b0010, respective to the value.
[11:8]	SHA1	Indicates support for SHA1 instructions in AArch64 state. Defined values are: **0b0000** 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	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[7:4]	AES	Indicates support for AES instructions in AArch64 state. Defined values are: **0b0000** 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. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	The reset values can be the following: 0b0000, 0b0010, respective to the value.
[3:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, ID_AA64ISAR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b000

Accessibility

MRS <Xt>, ID_AA64ISAR0_EL1

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

```
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR0_EL1;
```

A.4.13 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx 0001 xxxx 0001 0001 0001 0001 0000 0001 0010 0001 0001 0000 0101 0010


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-120: AArch64_id_aa64isar1_el1 bit assignments

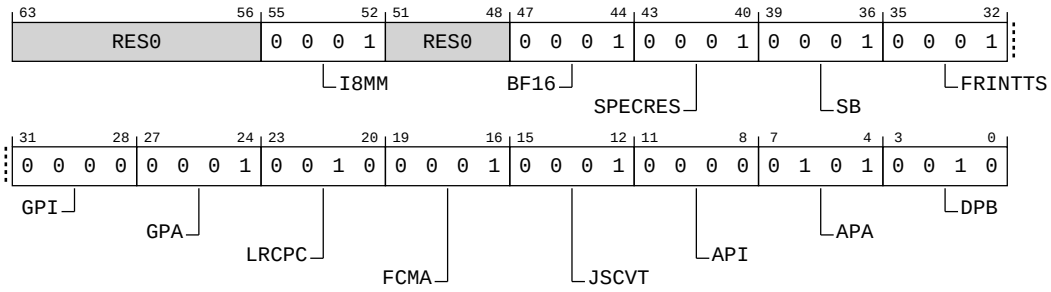


Table A-287: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:52]	I8MM	Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values are: **0b0001** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	0b0001
[51:48]	**RES0**	Reserved	**RES0**
[47:44]	BF16	Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are: **0b0001** BFCVT, BFCVTN, BFCVTN2, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	0b0001
[43:40]	SPECRES	Indicates support for prediction invalidation instructions in AArch64 state. Defined values are: **0b0001** CFP RCTX, DVP RCTX and CPP RCTX instructions are implemented.	0b0001
[39:36]	SB	Indicates support for SB instruction in AArch64 state. Defined values are: **0b0001** SB instruction is implemented.	0b0001
[35:32]	FRINTTS	Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are: **0b0001** FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented.	0b0001
[31:28]	GPI	Indicates support for an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0000** Generic Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	0b0000
[27:24]	GPA	Indicates whether the QARMA5 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0001** Generic Authentication using the QARMA5 algorithm is implemented. This includes the `PACGA` instruction.	0b0001

Bits	Name	Description	Reset
[23:20]	LRCPC	Indicates support for weaker release consistency, RCpc, based model. Defined values are: **0b0010** The LDAPR*, LDAPUR*, and STLUR* instructions are implemented.	0b0010
[19:16]	FCMA	Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are: **0b0001** The FCMLA and FCADD instructions are implemented.	0b0001
[15:12]	JSCVT	Indicates support for JavaScript conversion from double precision floating point values to integers in AArch64 state. Defined values are: **0b0001** The FJCVTZS instruction is implemented.	0b0001
[11:8]	API	Indicates whether an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: **0b0000** Address Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	0b0000
[7:4]	APA	Indicates whether the QARMA5 algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: **0b0101** Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.	0b0101
[3:0]	DPB	Data Persistence writeback. Indicates support for the DC CVAP and DC CVADP instructions in AArch64 state. Defined values are: **0b0010** DC CVAP and DC CVADP supported.	0b0010

Access

MRS <Xt>, ID_AA64ISAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b001

Accessibility

MRS <Xt>, ID_AA64ISAR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR1_EL1;
```

```
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR1_EL1;
```

A.4.14 ID_AA64ISAR2_EL1, AArch64 Instruction Set Attribute Register 2

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-121: AArch64_id_aa64isar2_el1 bit assignments

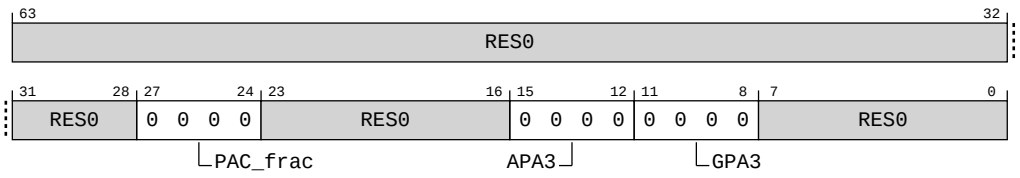


Table A-289: ID_AA64ISAR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:28]	**RES0**	Reserved	**RES0**
[27:24]	PAC_frac	Indicates whether the ConstPACField() function used as part of the PAC addition returns FALSE or TRUE. **0b0000** ConstPACField() returns FALSE.	0b0000
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	APA3	Indicates whether the QARMA3 algorithm is implemented in the PE for address authentication in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: **0b0000** Address Authentication using the QARMA3 algorithm is not implemented.	0b0000
[11:8]	GPA3	Indicates whether the QARMA3 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0000** Generic Authentication using the QARMA3 algorithm is not implemented.	0b0000
[7:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, ID_AA64ISAR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b010

Accessibility

MRS <Xt>, ID_AA64ISAR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    X[t, 64] = ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR2_EL1;
```

A.4.15 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx 0000 0010 0010 0010 0000 0000 0001 xxxx 0001 0001 0010 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-122: AArch64_id_aa64mmfr0_el1 bit assignments

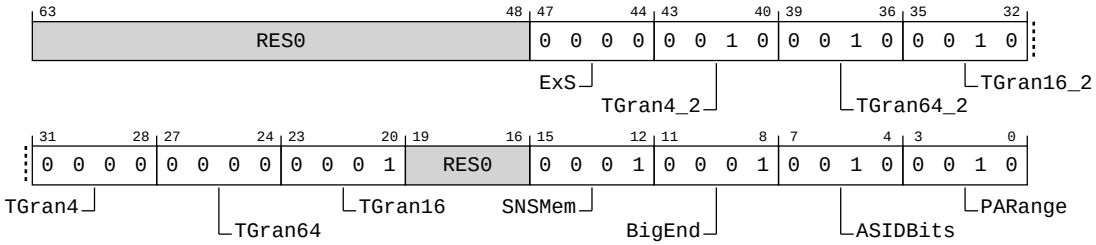


Table A-291: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:44]	ExS	Indicates support for disabling context synchronizing exception entry and exit. Defined values are: **0b0000** All exception entries and exits are context synchronization events. All other values are reserved. FEAT_ExS implements the functionality identified by the value 0b0001.	0b0000

Bits	Name	Description	Reset
[43:40]	TGran4_2	Indicates support for 4KB memory granule size at stage 2. Defined values are: **0b0010** 4KB granule supported at stage 2.	0b0010
[39:36]	TGran64_2	Indicates support for 64KB memory granule size at stage 2. Defined values are: **0b0010** 64KB granule supported at stage 2.	0b0010
[35:32]	TGran16_2	Indicates support for 16KB memory granule size at stage 2. Defined values are: **0b0010** 16KB granule supported at stage 2.	0b0010
[31:28]	TGran4	Indicates support for 4KB memory translation granule size. Defined values are: **0b0000** 4KB granule supported.	0b0000
[27:24]	TGran64	Indicates support for 64KB memory translation granule size. Defined values are: **0b0000** 64KB granule supported.	0b0000
[23:20]	TGran16	Indicates support for 16KB memory translation granule size. Defined values are: **0b0001** 16KB granule supported.	0b0001
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	SNSMem	Indicates support for a distinction between Secure and Non-secure Memory. Defined values are: **0b0001** Does support a distinction between Secure and Non-secure Memory.	0b0001
[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.	0b0001
[7:4]	ASIDBits	Number of ASID bits. Defined values are: **0b0010** 16 bits.	0b0010
[3:0]	PARange	Physical Address range supported. Defined values are: **0b0010** 40 bits, 1TB.	0b0010

Access

MRS <Xt>, ID_AA64MMFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b000

Accessibility

MRS <Xt>, ID_AA64MMFR0_EL1

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

```
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR0_EL1;
```

A.4.16 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0001 xxxx xxxx 0000 xxxx xxxx 0000 xxxx 0001 0000 0011 0001 0010 0001 0010 0010

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-123: AArch64_id_aa64mmfr1_el1 bit assignments

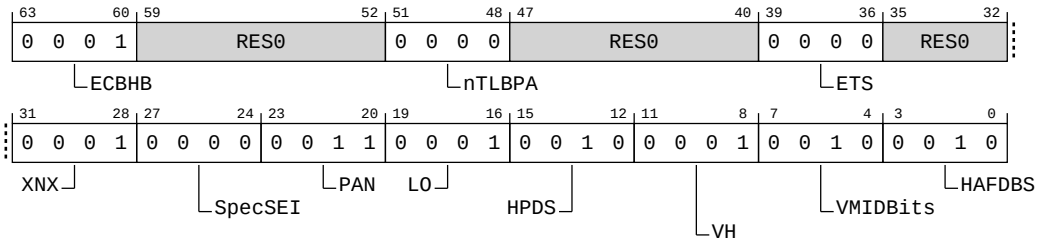


Table A-293: ID_AA64MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	ECBHB	Indicates support for cache maintenance instruction permission. Defined values are: **0b0001** The branch history information created in a context before an exception to a higher exception level using AArch64 cannot be used by code before that exception to exploitatively control the execution of any indirect branches in code in a different context after the exception.	0b0001
[59:52]	**RES0**	Reserved	**RES0**
[51:48]	nTLBPA	Indicates support for intermediate caching of translation table walks. Defined values are: **0b0000** The intermediate caching of translation table walks might include non-coherent physical translation caches.	0b0000
[47:40]	**RES0**	Reserved	**RES0**
[39:36]	ETS	Indicates support for Enhanced Translation Synchronization. Defined values are: **0b0000** Enhanced Translation Synchronization is not supported.	0b0000
[35:32]	**RES0**	Reserved	**RES0**
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are: **0b0001** Distinction between EL0 and EL1 execute-never control at stage 2 supported.	0b0001
[27:24]	SpecSEI	Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. **0b0000** The PE never generates an SError interrupt due to an External abort on a speculative read.	0b0000
[23:20]	PAN	Privileged Access Never. Indicates support for the PAN bit in PSTATE, AArch64-SPSR_EL1, AArch64-SPSR_EL2, AArch64-SPSR_EL3, and AArch64-DSPSR_EL0. Defined values are: **0b0011** PAN supported, AT S1E1RP and AT S1E1WP instructions supported, and AArch64-SCTLR_EL1.EPAN and AArch64-SCTLR_EL2.EPAN bits supported.	0b0011
[19:16]	LO	LORegions. Indicates support for LORegions. Defined values are: **0b0001** LORegions supported.	0b0001

Bits	Name	Description	Reset
[15:12]	HPDS	Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are: **0b0010** Disabling of hierarchical controls supported with the TCR_EL1.{HPD1, HPD0}, TCR_EL2.HPD or TCR_EL2.{HPD1, HPD0}, and TCR_EL3.HPD bits and adds possible hardware allocation of bits[62:59] of the translation table descriptors from the final lookup level for **IMPLEMENTATION DEFINED** use.	0b0010
[11:8]	VH	Virtualization Host Extensions. Defined values are: **0b0001** Virtualization Host Extensions supported.	0b0001
[7:4]	VMIDBits	Number of VMID bits. Defined values are: **0b0010** 16 bits	0b0010
[3:0]	HAFDBS	Hardware updates to Access flag and Dirty state in translation tables. Defined values are: **0b0010** Hardware update of both the Access flag and dirty state is supported.	0b0010

Access

MRS <Xt>, ID_AA64MMFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b001

Accessibility

MRS <Xt>, ID_AA64MMFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR1_EL1;
```

A.4.17 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0001 0010 0010 0001 xxxx 0001 0001 0001 0001 0000 0001 0000 0001 0000 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-124: AArch64_id_aa64mmfr2_el1 bit assignments

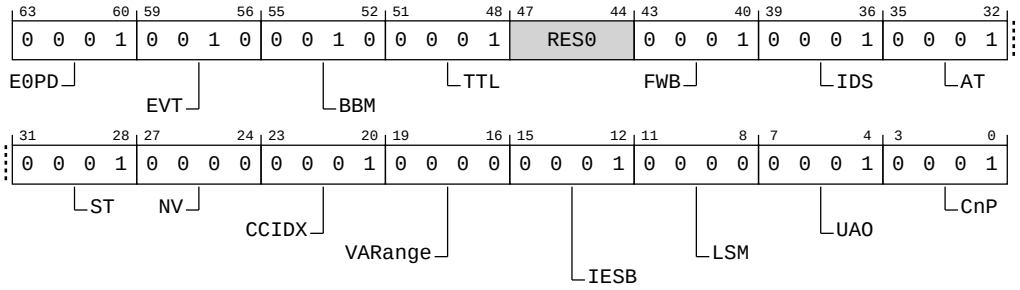


Table A-295: ID_AA64MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	E0PD	Indicates support for the E0PD mechanism. Defined values are: **0b0001** E0PDx mechanism is implemented.	0b0001

Bits	Name	Description	Reset
[59:56]	EVT	Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps. Defined values are: **0b0010** AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are supported.	0b0010
[55:52]	BBM	Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. **0b0010** Level 2 support for changing block size is supported.	0b0010
[51:48]	TTL	Indicates support for TTL field in address operations. Defined values are: **0b0001** TLB maintenance instructions by address have bits[47:44] holding the TTL field.	0b0001
[47:44]	**RES0**	Reserved	**RES0**
[43:40]	FWB	Indicates support for AArch64-HCR_EL2.FWB. Defined values are: **0b0001** AArch64-HCR_EL2.FWB is supported.	0b0001
[39:36]	IDS	Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. Defined values are: **0b0001** All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.	0b0001
[35:32]	AT	Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are: **0b0001** Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.	0b0001
[31:28]	ST	Identifies support for small translation tables. Defined values are: **0b0001** The maximum value of the TCR_ELx.{T0SZ,T1SZ} and VTCR_EL2.T0SZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules.	0b0001
[27:24]	NV	Nested Virtualization. If EL2 is implemented, indicates support for the use of nested virtualization. Defined values are: **0b0000** Nested virtualization is not supported.	0b0000
[23:20]	CCIDX	Support for the use of revised AArch64-CCSIDR_EL1 register format. Defined values are: **0b0001** 64-bit format implemented for all levels of the CCSIDR_EL1.	0b0001
[19:16]	VARange	Indicates support for a larger virtual address. Defined values are: **0b0000** VMSAv8-64 supports 48-bit VAs.	0b0000
[15:12]	IESB	Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are: **0b0001** IESB bit in the SCTLR_ELx registers is supported.	0b0001

Bits	Name	Description	Reset
[11:8]	LSM	Indicates support for LSMAOE and nTLSMD bits in AArch64-SCTLR_EL1 and AArch64-SCTLR_EL2. Defined values are: **0b0000** LSMAOE and nTLSMD bits not supported.	0b0000
[7:4]	UAO	User Access Override. Defined values are: **0b0001** UAO supported.	0b0001
[3:0]	CnP	Indicates support for Common not Private translations. Defined values are: **0b0001** Common not Private translations supported.	0b0001

Access

MRS <Xt>, ID_AA64MMFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b010

Accessibility

MRS <Xt>, ID_AA64MMFR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR2_EL1;
```

A.4.18 MPAMIDR_EL1, MPAM ID Register (EL1)

Indicates the presence and maximum PARTID and PMG values supported in the implementation. It also indicates whether the implementation supports MPAM virtualization.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0001 xxxx xxxx xxx0 011x 0000 0000 0011 1111

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

MPAMIDR_EL1 indicates the MPAM implementation parameters of the PE.

Figure A-125: AArch64_mpamidr_el1 bit assignments

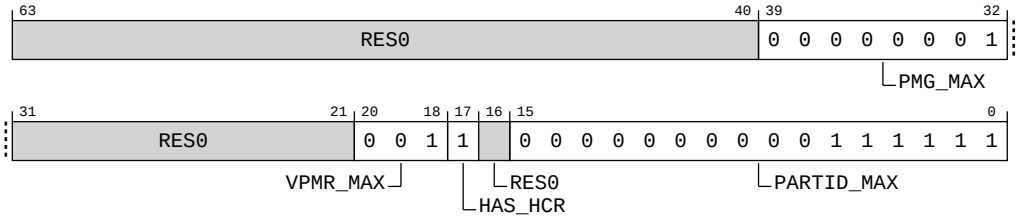


Table A-297: MPAMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:32]	PMG_MAX	The largest value of PMG that the implementation can generate. The PMG_I and PMG_D fields of every MPAMn_ELx must implement at least enough bits to represent PMG_MAX. **0b00000001** Max PMG field is 1 (1-bit)	0x01
[31:21]	**RES0**	Reserved	**RES0**
[20:18]	VPMR_MAX	Indicates the maximum register index n for the MPAMVPM<n>_EL2 registers. **0b001** 2 MPAMVPMn_EL2 registers are implemented	0b001
[17]	HAS_HCR	HAS_HCR indicates that the PE implementation supports MPAM virtualization, including AArch64-MPAMHCR_EL2, AArch64-MPAMVPMV_EL2, and MPAMVPM<n>_EL2 with n in the range 0 to VPMR_MAX. Must be 0 if EL2 is not implemented in either Security state. **0b1** MPAM virtualization is supported.	0b1
[16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[15:0]	PARTID_MAX	The largest value of PARTID that the implementation can generate. The PARTID_I and PARTID_D fields of every MPAMn_ELx must implement at least enough bits to represent PARTID_MAX. **0b0000000000111111** Max PARTID field is 63	0x003F

Access

MRS <Xt>, MPAMIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b100

Accessibility

MRS <Xt>, MPAMIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAMHCR_EL2.TRAP_MPAMIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MPAMIDR_EL1;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMIDR_EL1;
```

A.4.19 IMP_CPUCFR_EL1, CPU Configuration Register

This register provides configuration information for the core.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-126: AArch64_imp_cpucfr_el1 bit assignments

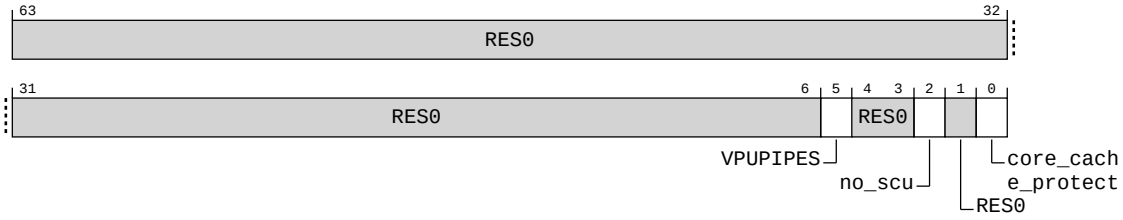


Table A-299: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:6]	**RES0**	Reserved	**RES0**
[5]	VPUPIPES	Indicates the number of Vector Processing Unit (VPU) pipes. Possible values of this bit are: **0b0** 2 x 128-bit **0b1** 4 x 128-bit	x
[4:3]	**RES0**	Reserved	**RES0**
[2]	no_scu	Indicates whether the SCU is present or not. Possible values of this bit are: **0b0** The SCU is present. **0b1** The SCU is not present.	x
[1]	**RES0**	Reserved	**RES0**
[0]	core_cache_protect	Indicates whether ECC is present or not. Possible values of this field are: **0b0** ECC is not present. **0b1** ECC is present.	x

Access

MRS <Xt>, S3_0_C15_C0_0

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b000

Accessibility

MRS <Xt>, S3_0_C15_C0_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUCFR_EL1;
```

A.4.20 CLIDR_EL1, Cache Level ID Register

Identifies the type of cache, or caches, that are implemented at each level. The register can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx x000 0000 0xx1 010x xx00 0xxx 0000 0000 0000 000x xx10 0011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-127: AArch64_clidr_el1 bit assignments

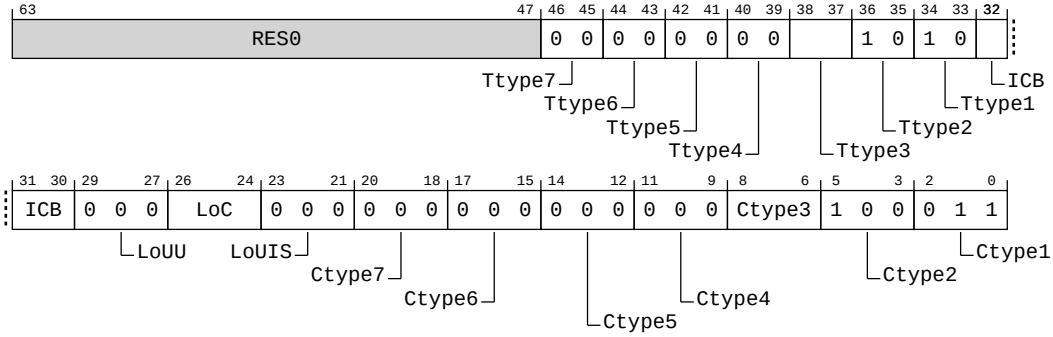


Table A-301: CLIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:47]	**RES0**	Reserved	**RES0**
[46:45]	Ttype7	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00
[44:43]	Ttype6	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00
[42:41]	Ttype5	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00
[40:39]	Ttype4	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00

Bits	Name	Description	Reset
[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	xx
[36:35]	Ttype2	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines.	0b10
[34:33]	Ttype1	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b10** Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines.	0b10
[32:30]	ICB	Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions. **0b010** When no L3 present, Level 2 **0b011** When L3 present, Level 3	The reset values can be the following: 0b010, 0b011, respective to the value.
[29:27]	LoUU	Level of Unification Uniprocessor for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **Note:** When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches. **0b000** Level of Unification Uniprocessor is before the L1 data cache.	0b000
[26:24]	LoC	Level of Coherence for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **0b010** When no L3 present, Level 2 **0b011** When L3 present, Level 3	xxx

Bits	Name	Description	Reset
[23:21]	LoUIS	Level of Unification Inner Shareable for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **Note:** When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches. **0b000** No cache level needs cleaning to Point of Unification	0b000
[20:18]	Ctype7	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[17:15]	Ctype6	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[14:12]	Ctype5	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[11:9]	Ctype4	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[8:6]	Ctype3	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No L3. **0b100** Unified instruction and data caches at L3	The reset values can be the following: 0b000, 0b100, respective to the value.
[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	0b100

Bits	Name	Description	Reset
[2:0]	Ctype1	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b011** Separate instruction and data caches at L1	0b011

Access

MRS <Xt>, CLIDR_EL1

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

Accessibility

MRS <Xt>, CLIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CLIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CLIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CLIDR_EL1;
```

A.4.21 GMID_EL1, Multiple tag transfer ID register

Indicates the block size that is accessed by the LDGM and STGM System instructions.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-128: AArch64_gmid_el1 bit assignments

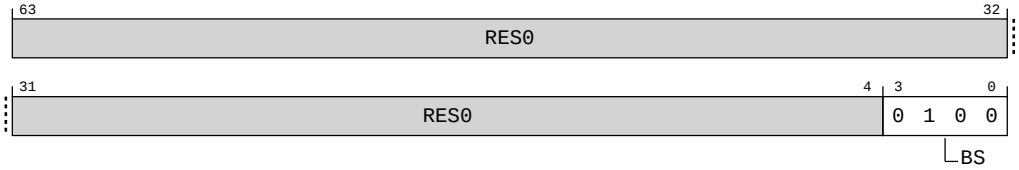


Table A-303: GMID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	**RES0**	Reserved	**RES0**
[3:0]	BS	Log_2 of the block size in words. The minimum supported size is 16B (value == 2) and the maximum is 256B (value == 6). **0b0100** Log2 of the block size is 4	0b0100

Access

MRS <Xt>, GMID_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	0b0000	0b0000	0b100

Accessibility

MRS <Xt>, GMID_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID5 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = GMID_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = GMID_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = GMID_EL1;
```

A.4.22 CTR_EL0, Cache Type Register

Provides information about the architecture of the caches.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xx00 0100 xx01 0100 0100 0100 11xx xxxx xxxx 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-129: AArch64_ctr_el0 bit assignments

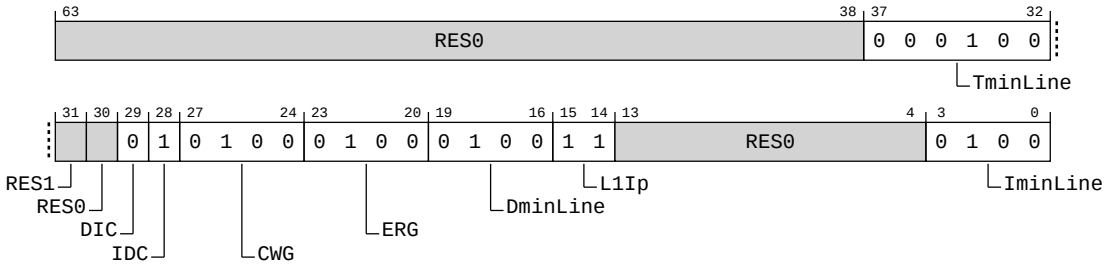


Table A-305: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:38]	**RES0**	Reserved	**RES0**
[37:32]	TminLine	Tag minimum Line. Log_2 of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE. **0b000100** Log2 of number of words (64/4=16) covered by Allocation Tags in the smallest cache line of all caches	0b000100

Bits	Name	Description	Reset
[31]	**RES1**	Reserved	**RES1**
[30]	**RES0**	Reserved	**RES0**
[29]	DIC	Instruction cache invalidation requirements for data to instruction coherence. **0b0** Instruction cache invalidation to the point of unification is required for instruction to data coherence.	0b0
[28]	IDC	Data cache clean requirements for instruction to data coherence. The meaning of this bit is: **0b1** Data cache clean to the Point of Unification is not required for instruction to data coherence.	0b1
[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.	0b0100
[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.	0b0100
[19:16]	DminLine	Log_2 of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE. **0b0100** 64 bytes.	0b0100
[15:14]	L1Ip	Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are: **0b11** Physical Index, Physical Tag (PIPT).	0b11
[13:4]	**RES0**	Reserved	**RES0**
[3:0]	IminLine	Log_2 of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE. **0b0100** 64 bytes.	0b0100

Access

MRS <Xt>, CTR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b0000	0b0000	0b001

Accessibility

MRS <Xt>, CTR_EL0

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

```
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CTR_EL0;
```

A.4.23 DCZID_EL0, Data Cache Zero ID register

Indicates the block size that is written with byte values of 0 by the DC ZVA (Data Cache Zero by Address) System instruction.

If FEAT_MTE is implemented, this register also indicates the granularity at which the DC GVA and DC GZVA instructions write.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0100


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-130: AArch64_dczid_el0 bit assignments

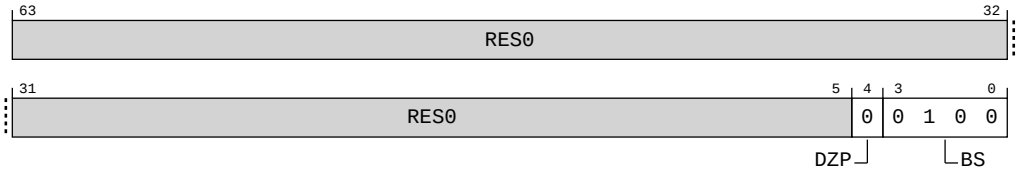


Table A-307: DCZID_EL0 bit descriptions

Bits	Name	Description	Reset
[63:5]	**RES0**	Reserved	**RES0**
[4]	DZP	Data Zero Prohibited. This field indicates whether use of DC ZVA instructions is permitted or prohibited. If FEAT_MTE is implemented, this field also indicates whether use of the DC GVA and DC GZVA instructions are permitted or prohibited. **0b0** Instructions are permitted.	0b0
[3:0]	BS	Log_2 of the block size in words. The maximum size supported is 2KB (value == 9). **0b0100** Log2 of the block size is 4	0b0100

Access

MRS <Xt>, DCZID_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, DCZID_EL0

```
if PSTATE.EL == EL0 then
    X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL1 then
    X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = DCZID_EL0;
```

A.5 AArch64 Special-purpose registers summary

The summary table provides an overview of all Special-purpose registers in the core. For more information about a register, click the register name in the table.

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

Table A-309: Special-purpose registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SPSR_EL1	3	0	C4	C0	0	–	64-bit	Saved Program Status Register (EL1)
ELR_EL1	3	0	C4	C0	1	–	64-bit	Exception Link Register (EL1)
SP_EL0	3	0	C4	C1	0	–	64-bit	Stack Pointer (EL0)
DSPSR_EL0	3	3	C4	C5	0	–	64-bit	Debug Saved Program Status Register
DLR_EL0	3	3	C4	C5	1	–	64-bit	Debug Link Register
SPSR_EL2	3	4	C4	C0	0	–	64-bit	Saved Program Status Register (EL2)
ELR_EL2	3	4	C4	C0	1	–	64-bit	Exception Link Register (EL2)
SP_EL1	3	4	C4	C1	0	–	64-bit	Stack Pointer (EL1)
SPSR_irq	3	4	C4	C3	0	–	64-bit	Saved Program Status Register (IRQ mode)
SPSR_abt	3	4	C4	C3	1	–	64-bit	Saved Program Status Register (Abort mode)
SPSR_und	3	4	C4	C3	2	–	64-bit	Saved Program Status Register (Undefined mode)
SPSR_fiq	3	4	C4	C3	3	–	64-bit	Saved Program Status Register (FIQ mode)
SPSR_EL3	3	6	C4	C0	0	–	64-bit	Saved Program Status Register (EL3)
ELR_EL3	3	6	C4	C0	1	–	64-bit	Exception Link Register (EL3)
SP_EL2	3	6	C4	C1	0	–	64-bit	Stack Pointer (EL2)

A.6 AArch64 Performance Monitors registers summary

The summary table provides an overview of all Performance Monitors registers in the core. For more information about a register, click the register name in the table.

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

Table A-310: Performance Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMINTENSET_EL1	3	0	C9	C14	1	–	64-bit	Performance Monitors Interrupt Enable Set register
PMINTENCLR_EL1	3	0	C9	C14	2	–	64-bit	Performance Monitors Interrupt Enable Clear register
PMMIR_EL1	3	0	C9	C14	6	–	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	3	C9	C12	0	–	64-bit	Performance Monitors Control Register
PMCNTENSET_EL0	3	3	C9	C12	1	–	64-bit	Performance Monitors Count Enable Set register
PMCNTENCLR_EL0	3	3	C9	C12	2	–	64-bit	Performance Monitors Count Enable Clear register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMOVSCLR_EL0	3	3	C9	C12	3	–	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_EL0	3	3	C9	C12	4	–	64-bit	Performance Monitors Software Increment register
PMSELR_EL0	3	3	C9	C12	5	–	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	3	3	C9	C12	6	–	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	3	C9	C12	7	–	64-bit	Performance Monitors Common Event Identification register 1
PMCCNTR_EL0	3	3	C9	C13	0	–	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	–	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	–	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	3	3	C9	C14	0	–	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	–	64-bit	Performance Monitors Overflow Flag Status Set register
PMEVCNTR0_EL0	3	3	C14	C8	0	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	–	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	–	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	–	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	–	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	–	64-bit	Performance Monitors Cycle Count Filter Register

A.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 registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 1000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-131: AArch64_pmmir_el1 bit assignments

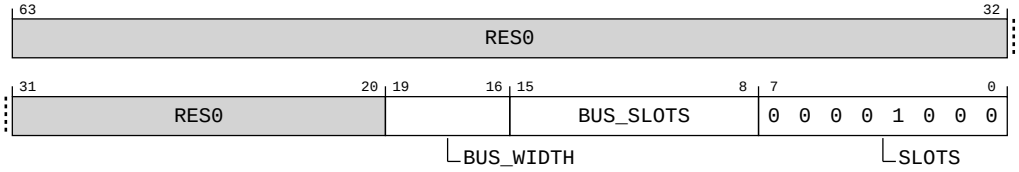


Table A-311: PMMIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[19:16]	BUS_WIDTH	Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as Log_2(number of bytes), plus one. **0b0000** The information is not available. **0b0011** Four bytes. **0b0100** 8 bytes. **0b0101** 16 bytes. **0b0110** 32 bytes. **0b0111** 64 bytes. **0b1000** 128 bytes. **0b1001** 256 bytes. **0b1010** 512 bytes. **0b1011** 1024 bytes. **0b1100** 2048 bytes. All other values are reserved. Each transfer is up to this number of bytes. An access might be smaller than the bus width. When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.	The reset values can be the following: `0b0000`, `0b0011`, `0b0100`, `0b0101`, `0b0110`, `0b0111`, `0b1000`, `0b1001`, `0b1010`, `0b1011`, `0b1100`, respective to the value.
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS. If the information is not available, this field will read as zero.	`8{x}`

Bits	Name	Description	Reset
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. **0b00001000** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 8.	0x08

Access

MRS <Xt>, PMMIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b110

Accessibility

MRS <Xt>, PMMIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMMIR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMMIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMMIR_EL1;
```

A.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

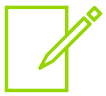
Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 x000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-132: AArch64_pmcr_el0 bit assignments

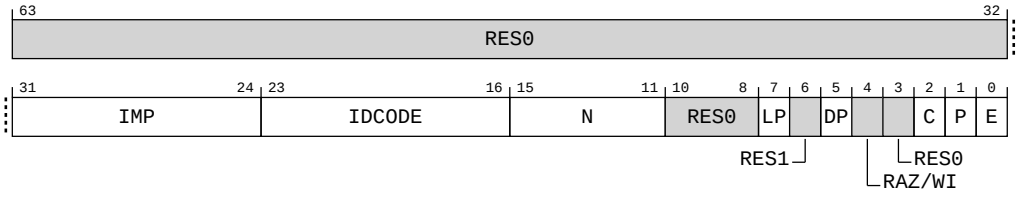


Table A-313: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	IMP	Implementer code. If this field is zero, then PMCR_EL0.IDCODE is **RES0** and software must use AArch64-MIDR_EL1 to identify the PE. Otherwise, this field and PMCR_EL0.IDCODE identify the PMU implementation to software. The implementer codes are allocated by Arm. A non-zero value has the same interpretation as AArch64-MIDR_EL1.Implementer. Use of this field is deprecated.	8{x}
[23:16]	IDCODE	**When AArch64-PMCR_EL0.IMP != '00000000'** Identification code. Use of this field is deprecated. Each implementer must maintain a list of identification codes that are specific to the implementer. A specific implementation is identified by the combination of the implementer code and the identification code. **Otherwise** RES0	8{x}

Bits	Name	Description	Reset
[15:11]	N	Number of event counters: **0b00110** When configured for 6 PMU counters, denotes 6 PMU counters implemented **0b10100** When configured for 20 PMU counters, denotes 20 PMU counters implemented	The reset values can be the following: 0b00110, 0b10100, respective to the value.
[10:8]	**RES0**	Reserved	**RES0**
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by an event counter overflow bit. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_EL0[31:0]. **0b1** Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_EL0[63:0]. If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)]. This field does not affect the operation of other event counters and AArch64-PMCCNTR_EL0. The operation of this field applies even when EL2 is disabled in the current Security state.	x
[6]	**RES1**	Reserved	**RES1**
[5]	DP	Disable cycle counter when event counting is prohibited. **0b0** Cycle counting by AArch64-PMCCNTR_EL0 is not affected by this mechanism. **0b1** Cycle counting by AArch64-PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen: • If FEAT_PMUv3p1 is implemented, EL2 is implemented, and AArch64-MDCR_EL2.HPMD is 1, then cycle counting by AArch64-PMCCNTR_EL0 is disabled at EL2. • If EL3 is implemented, AArch64-MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or AArch64-MDCR_EL3.MPMX is 0, then cycle counting by AArch64-PMCCNTR_EL0 is disabled at EL3 and in Secure state. If AArch64-MDCR_EL2.HPMN is not 0, this is when event counting by event counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited or frozen.	x
[4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[2]	C	Cycle counter reset. The effects of writing to this bit are: **0b0** No action. **0b1** Reset AArch64-PMCCNTR_EL0 to zero. Resetting AArch64-PMCCNTR_EL0 does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR_EL0.LC is ignored, and bits [63:0] of the cycle counter are reset. Access to this field is: WO/**RAZ**	0b0
[1]	P	Event counter reset. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. **0b0** No action. **0b1** If n is in the range of affected event counters, resets each event counter AArch64-PMEVCNTR<n>_EL0 to zero. The effects of writing to this bit are: • If EL2 is implemented and enabled in the current Security state, in EL0 and EL1, if PMN is not 0, a write of 1 to this bit resets event counters in the range [0 .. (PMN-1)]. • If EL2 is disabled in the current Security state, a write of 1 to this bit resets all the event counters. • In EL2 and EL3, a write of 1 to this bit resets all the event counters. • This field does not affect the operation of other event counters and AArch64-PMCCNTR_EL0. Resetting the event counters does not change the event counter overflow bits. If FEAT_PMUv3p5 is implemented, the values of AArch64-MDCR_EL2.HLP and PMCR_EL0.LP are ignored, and bits [63:0] of all affected event counters are reset. Access to this field is: WO/**RAZ**	0b0

Bits	Name	Description	Reset
[0]	E	Enable. If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. **0b0** AArch64-PMCCNTR_EL0 is disabled and event counters AArch64-PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are disabled. **0b1** AArch64-PMCCNTR_EL0 and event counters AArch64-PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are enabled by AArch64-PMCNTENSET_EL0. If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)]. This field does not affect the operation of other event counters. The operation of this field applies even when EL2 is disabled in the current Security state.	0b0

Access

MRS <Xt>, PMCR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

MSR PMCR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

Accessibility

MRS <Xt>, PMCR_EL0

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCR_EL0;
```

MSR PMCR_EL0, <Xt>

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMCR_EL0 = X[t, 64];
```

A.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.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

0000 1111 0000 1111 0001 1010 0111 1111 0111 1111 1111 1111 0000 1111 0011 1111

Bit descriptions

Figure A-133: AArch64_pmceid0_el0 bit assignments

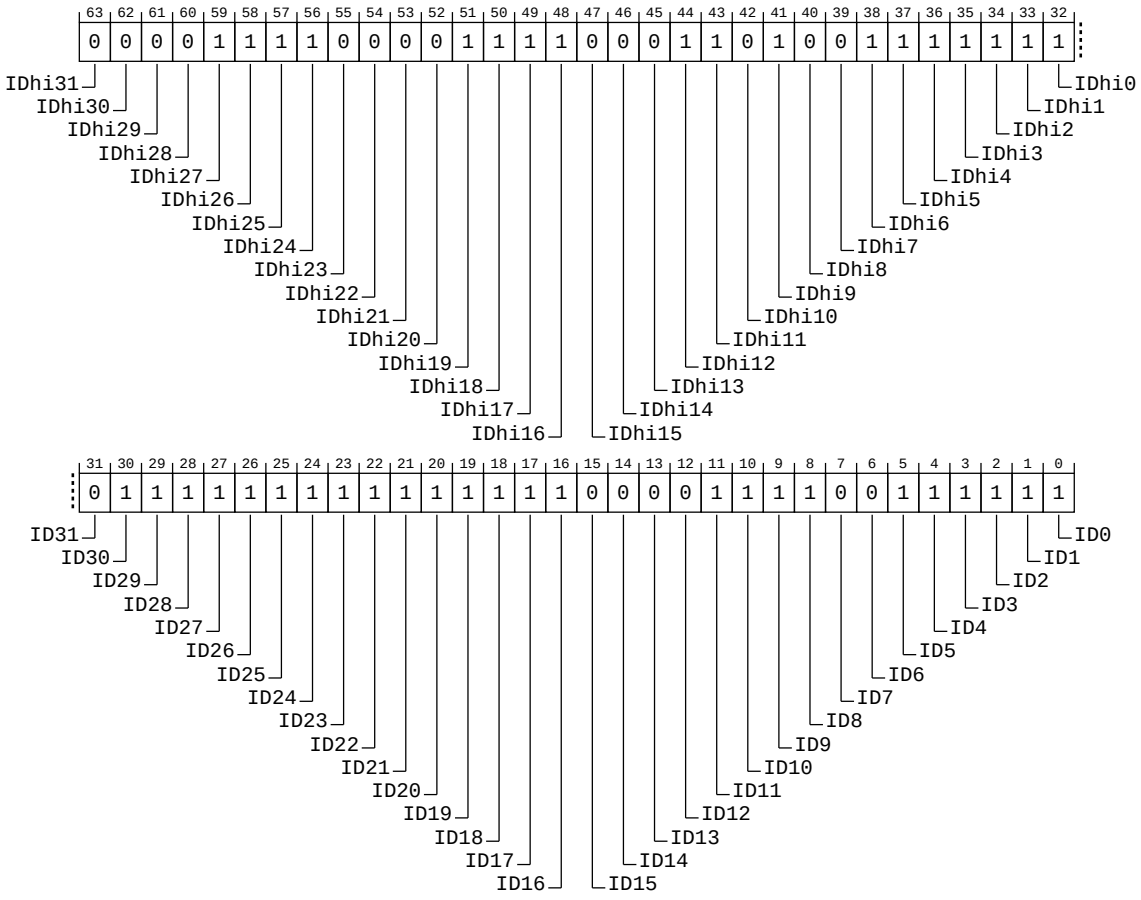


Table A-316: PMCEID0_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) **0b0** The Common event is not implemented, or not counted.	0b0
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) **0b0** The Common event is not implemented, or not counted.	0b0
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) **0b0** The Common event is not implemented, or not counted.	0b0
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 **0b1** The Common event is implemented.	0b1
[58]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 **0b1** The Common event is implemented.	0b1
[57]	IDhi25	IDhi25 corresponds to common event (0x4019) CTI_TRIGOUT5 **0b1** The Common event is implemented.	0b1
[56]	IDhi24	IDhi24 corresponds to common event (0x4018) CTI_TRIGOUT4 **0b1** The Common event is implemented.	0b1
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4017) **0b0** The Common event is not implemented, or not counted.	0b0
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4016) **0b0** The Common event is not implemented, or not counted.	0b0
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4015) **0b0** The Common event is not implemented, or not counted.	0b0
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4014) **0b0** The Common event is not implemented, or not counted.	0b0
[51]	IDhi19	IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 **0b1** The Common event is implemented.	0b1
[50]	IDhi18	IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 **0b1** The Common event is implemented.	0b1
[49]	IDhi17	IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 **0b1** The Common event is implemented.	0b1
[48]	IDhi16	IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 **0b1** The Common event is implemented.	0b1
[47]	IDhi15	IDhi15 corresponds to common event (0x400f) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[46]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[45]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS **0b0** The Common event is not implemented, or not counted.	0b0
[44]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP **0b1** The Common event is implemented.	0b1
[43]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0b1** The Common event is implemented.	0b1
[42]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0b0** The Common event is not implemented, or not counted.	0b0
[41]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0b1** The Common event is implemented.	0b1
[40]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[39]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[38]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS **0b1** The Common event is implemented.	0b1
[37]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM **0b1** The Common event is implemented.	0b1
[36]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0b1** The Common event is implemented.	0b1
[35]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0b1** The Common event is implemented.	0b1
[34]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0b1** The Common event is implemented.	0b1
[33]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0b1** The Common event is implemented.	0b1
[32]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0b1** The Common event is implemented.	0b1

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.	0b0
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN **0b1** The Common event is implemented.	0b1
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES **0b1** The Common event is implemented.	0b1
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED **0b1** The Common event is implemented.	0b1
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC **0b1** The Common event is implemented.	0b1
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR **0b1** The Common event is implemented.	0b1
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS **0b1** The Common event is implemented.	0b1
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB **0b1** The Common event is implemented.	0b1
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0b1** The Common event is implemented.	0b1
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB **0b1** The Common event is implemented.	0b1
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE **0b1** The Common event is implemented.	0b1
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS **0b1** The Common event is implemented.	0b1
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES **0b1** The Common event is implemented.	0b1
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED **0b1** The Common event is implemented.	0b1
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **0b1** The Common event is implemented.	0b1
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **0b1** The Common event is implemented.	0b1
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN **0b1** The Common event is implemented.	0b1
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED **0b1** The Common event is implemented.	0b1
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL **0b1** The Common event is implemented.	0b1
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL **0b1** The Common event is implemented.	0b1
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR **0b1** The Common event is implemented.	0b1

Access

MRS <Xt>, PMCEID0_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b110

Accessibility

MRS <Xt>, PMCEID0_EL0

```
if PSTATE.EL == EL0 then
    if 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID0_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID0_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID0_EL0;
```

```
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCEID0_EL0;
```

A.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.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0111 0111 1111 1110 1111 0010 1010 1110 0111 1111

Bit descriptions

Figure A-134: AArch64_pmceid1_el0 bit assignments

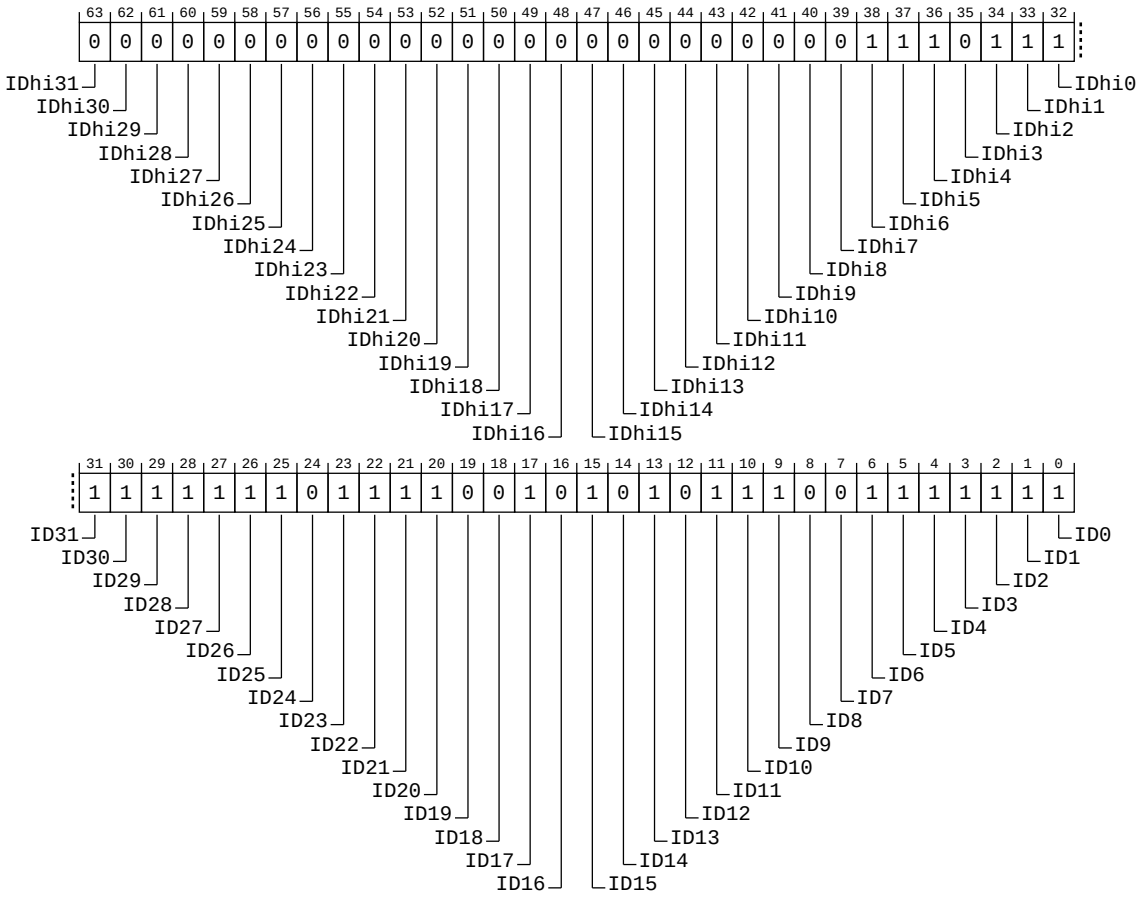


Table A-318: PMCEID1_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) **0b0** The Common event is not implemented, or not counted.	0b0
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) **0b0** The Common event is not implemented, or not counted.	0b0
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) **0b0** The Common event is not implemented, or not counted.	0b0
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) **0b0** The Common event is not implemented, or not counted.	0b0
[58]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) **0b0** The Common event is not implemented, or not counted.	0b0
[57]	IDhi25	IDhi25 corresponds to a Reserved Event event (0x4039) **0b0** The Common event is not implemented, or not counted.	0b0
[56]	IDhi24	IDhi24 corresponds to a Reserved Event event (0x4038) **0b0** The Common event is not implemented, or not counted.	0b0
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) **0b0** The Common event is not implemented, or not counted.	0b0
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) **0b0** The Common event is not implemented, or not counted.	0b0
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) **0b0** The Common event is not implemented, or not counted.	0b0
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) **0b0** The Common event is not implemented, or not counted.	0b0
[51]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) **0b0** The Common event is not implemented, or not counted.	0b0
[50]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) **0b0** The Common event is not implemented, or not counted.	0b0
[49]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) **0b0** The Common event is not implemented, or not counted.	0b0
[48]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) **0b0** The Common event is not implemented, or not counted.	0b0
[47]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **0b0** The Common event is not implemented, or not counted.	0b0
[46]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[45]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **0b0** The Common event is not implemented, or not counted.	0b0
[44]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **0b0** The Common event is not implemented, or not counted.	0b0
[43]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **0b0** The Common event is not implemented, or not counted.	0b0
[42]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **0b0** The Common event is not implemented, or not counted.	0b0
[41]	IDhi9	IDhi9 corresponds to a Reserved Event event (0x4029) **0b0** The Common event is not implemented, or not counted.	0b0
[40]	IDhi8	IDhi8 corresponds to a Reserved Event event (0x4028) **0b0** The Common event is not implemented, or not counted.	0b0
[39]	IDhi7	IDhi7 corresponds to a Reserved Event event (0x4027) **0b0** The Common event is not implemented, or not counted.	0b0
[38]	IDhi6	IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR **0b1** The Common event is implemented.	0b1
[37]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD **0b1** The Common event is implemented.	0b1
[36]	IDhi4	IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED **0b1** The Common event is implemented.	0b1
[35]	IDhi3	IDhi3 corresponds to common event (0x4023) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[34]	IDhi2	IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT **0b1** The Common event is implemented.	0b1
[33]	IDhi1	IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT **0b1** The Common event is implemented.	0b1
[32]	IDhi0	IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **0b1** The Common event is implemented.	0b1
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **0b1** The Common event is implemented.	0b1
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **0b1** The Common event is implemented.	0b1
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **0b1** The Common event is implemented.	0b1
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **0b1** The Common event is implemented.	0b1
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **0b1** The Common event is implemented.	0b1
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD **0b1** The Common event is implemented.	0b1
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD **0b0** The Common event is not implemented, or not counted.	0b0
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD **0b1** The Common event is implemented.	0b1
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD **0b1** The Common event is implemented.	0b1
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WALK **0b1** The Common event is implemented.	0b1
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WALK **0b1** The Common event is implemented.	0b1
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) **0b0** The Common event is not implemented, or not counted.	0b0
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS **0b1** The Common event is implemented.	0b1
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB **0b0** The Common event is not implemented, or not counted.	0b0
[15]	ID15	ID15 corresponds to common event (0x2f) L2D_TLB **0b1** The Common event is implemented.	0b1
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **0b0** The Common event is not implemented, or not counted.	0b0
[13]	ID13	ID13 corresponds to common event (0x2d) L2D_TLB_REFILL **0b1** The Common event is implemented.	0b1
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **0b1** The Common event is implemented.	0b1
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0b1** The Common event is implemented.	0b1
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL **0b0** The Common event is not implemented, or not counted.	0b0
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB **0b1** The Common event is implemented.	0b1
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB **0b1** The Common event is implemented.	0b1
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND **0b1** The Common event is implemented.	0b1
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED **0b1** The Common event is implemented.	0b1
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED **0b1** The Common event is implemented.	0b1
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE **0b1** The Common event is implemented.	0b1

Access

MRS <Xt>, PMCEID1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b111

Accessibility

MRS <Xt>, PMCEID1_EL0

```
if PSTATE.EL == EL0 then
    if 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID1_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID1_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID1_EL0;
```

```
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCEID1_EL0;
```

A.6.5 PMEVCNTR0_EL0, Performance Monitors Event Count Registers

Holds performance monitors event counter 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-135: AArch64_pmevcntr0_el0 bit assignments

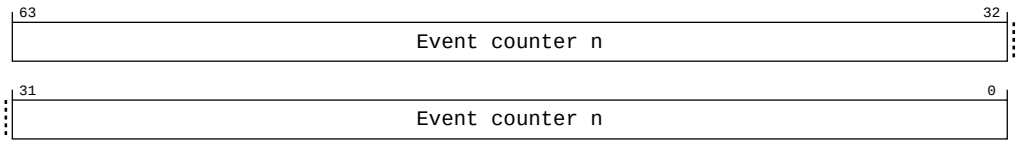


Table A-320: PMEVCNTR0_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.	64{x}

Access

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVCNTR0_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b000

MSR PMEVCNTR0_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b000

Accessibility

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.

- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVCNTR0_EL0

```
if 0 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' 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() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[0];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[0];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTR_EL0[0];
```

MSR PMEVCNTR0_EL0, <Xt>

```
if 0 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[0] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[0] = X[t, 64];
```

A.6.6 PMEVCNTR1_EL0, Performance Monitors Event Count Registers

Holds performance monitors event counter 1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-136: AArch64_pmevcntr1_el0 bit assignments

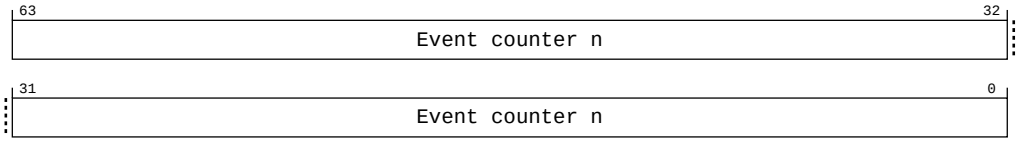


Table A-323: PMEVCNTR1_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.	64{x}

Access

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVCNTR1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b001

MSR PMEVCNTR1_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b001

Accessibility

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of

accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVCNTR1_EL0

```
if 1 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' 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() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[1];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[1];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTR_EL0[1];
```

MSR PMEVCNTR1_EL0, <Xt>

```
if 1 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        PMEVCNTR_EL0[1] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[1] = X[t, 64];
```

A.6.7 PMEVCNTR2_EL0, Performance Monitors Event Count Registers

Holds performance monitors event counter 2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-137: AArch64_pmevcntr2_el0 bit assignments

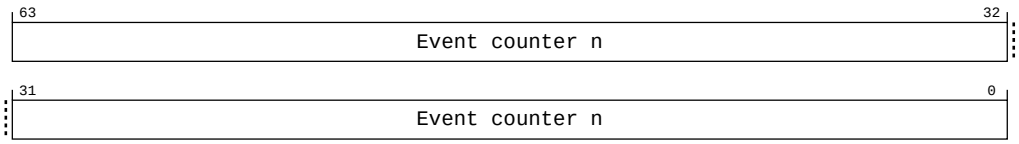


Table A-326: PMEVCNTR2_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.	64{x}

Access

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVCNTR2_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b010

MSR PMEVCNTR2_EL0, <Xt>

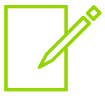
op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b010

Accessibility

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVCNTR2_EL0

```
if 2 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' 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() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[2];
```

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[2];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[2];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTR_EL0[2];
```

MSR PMEVCNTR2_EL0, <Xt>

```
if 2 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[2] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[2] = X[t, 64];
```

A.6.8 PMEVCNTR3_EL0, Performance Monitors Event Count Registers

Holds performance monitors event counter 3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-138: AArch64_pmevcntr3_el0 bit assignments

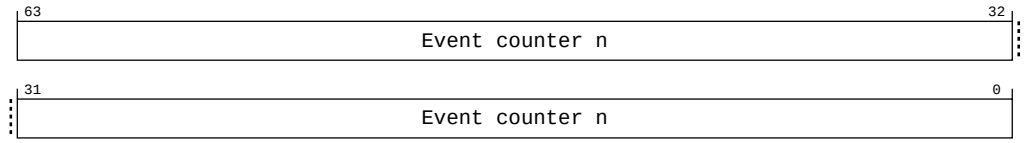


Table A-329: PMEVCNTR3_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.	64{x}

Access

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVCNTR3_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b011

MSR PMEVCNTR3_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b011

Accessibility

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVCNTR3_EL0

```
if 3 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' 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() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[3];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[3];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTR_EL0[3];
```

MSR PMEVCNTR3_EL0, <Xt>

```
if 3 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[3] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[3] = X[t, 64];
```

A.6.9 PMEVCNTR4_EL0, Performance Monitors Event Count Registers

Holds performance monitors event counter 4.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-139: AArch64_pmevcntr4_el0 bit assignments

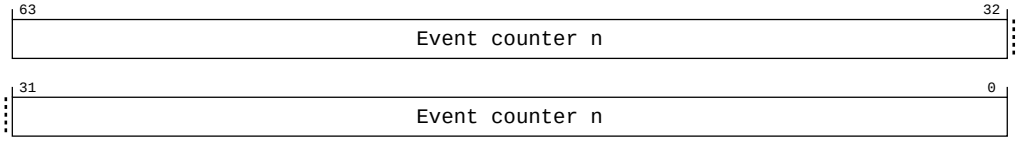


Table A-332: PMEVCNTR4_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.	64{x}

Access

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVCNTR4_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b100

MSR PMEVCNTR4_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b100

Accessibility

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVCNTR4_EL0

```
if 4 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' 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() && 4 >= AArch64.GetNumEventCountersAccessible() then
```

```
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[4];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[4];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[4];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVCNTR_EL0[4];
```

MSR PMEVCNTR4_EL0, <Xt>

```
if 4 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 4 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[4] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[4] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[4] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[4] = X[t, 64];
```

A.6.10 PMEVCNTR5_EL0, Performance Monitors Event Count Registers

Holds performance monitors event counter 5.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-140: AArch64_pmevcntr5_el0 bit assignments

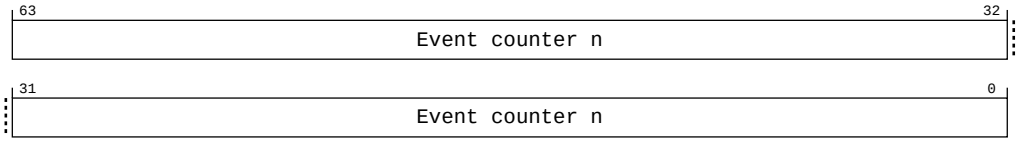


Table A-335: PMEVCNTR5_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.	`64{x}`

Access

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVCNTR5_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b101

MSR PMEVCNTR5_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1000	0b101

Accessibility

PMEVCNTR<n>_EL0 can also be accessed by using AArch64-PMXEVCNTR_EL0 with AArch64-PMSELR_EL0.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.

- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVCNTR5_EL0

```
if 5 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' 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() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[5];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[5];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVCNTR_EL0[5];
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMEVCNTR_EL0[5];
```

MSR PMEVCNTR5_EL0, <Xt>

```
if 5 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[5] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[5] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVCNTR_EL0[5] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVCNTR_EL0[5] = X[t, 64];
```

A.6.11 PMEVTYPER0_EL0, Performance Monitors Event Type Registers

Configures event counter 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-141: AArch64_pmevtyper0_el0 bit assignments

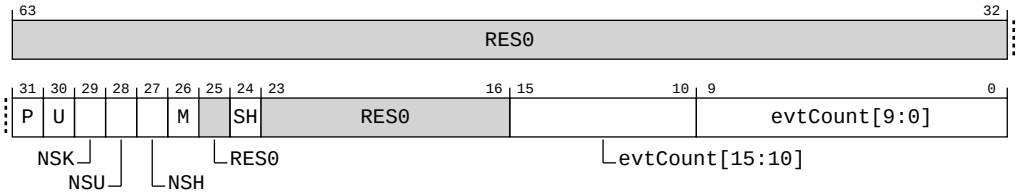


Table A-338: PMEVTYPER0_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter AArch64-PMEVCNTR<n>_EL0. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVTYPER0_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b000

MSR PMEVTYPER0_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b000

Accessibility

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.

- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVTYPER0_EL0

```
if 0 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[0];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[0];
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMEVTYPER_EL0[0];
```

MSR PMEVTYPER0_EL0, <Xt>

```
if 0 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[0] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[0] = X[t, 64];
```

A.6.12 PMEVTYPER1_EL0, Performance Monitors Event Type Registers

Configures event counter 1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-142: AArch64_pmevtyper1_el0 bit assignments

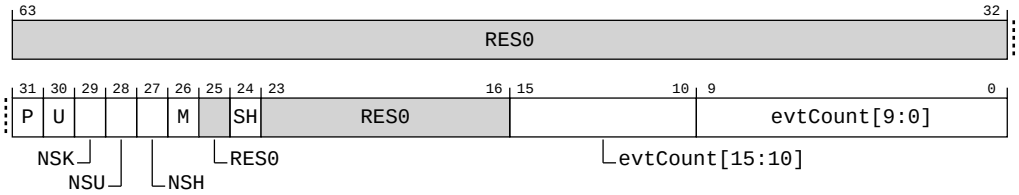


Table A-341: PMEVTYPER1_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter AArch64-PMEVCNTR<n>_EL0. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVTYPER1_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b001

MSR PMEVTYPER1_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b001

Accessibility

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.

- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVTYPER1_EL0

```
if 1 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[1];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[1];
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMEVTYPER_EL0[1];
```

MSR PMEVTYPER1_EL0, <Xt>

```
if 1 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[1] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[1] = X[t, 64];
```

A.6.13 PMEVTYPER2_EL0, Performance Monitors Event Type Registers

Configures event counter 2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-143: AArch64_pmevtyper2_el0 bit assignments

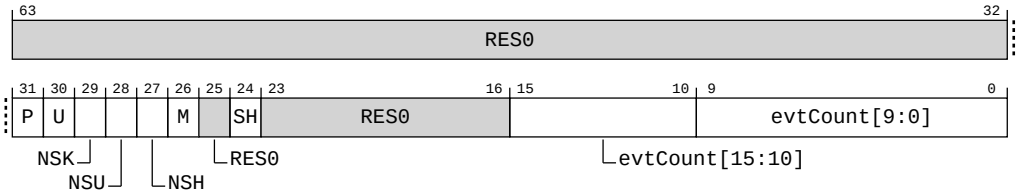


Table A-344: PMEVTYPER2_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter AArch64-PMEVCNTR<n>_EL0. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVTYPER2_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b010

MSR PMEVTYPER2_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b010

Accessibility

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.

- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVTYPER2_EL0

```
if 2 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[2];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[2];
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMEVTYPER_EL0[2];
```

MSR PMEVTYPER2_EL0, <Xt>

```
if 2 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[2] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[2] = X[t, 64];
```

A.6.14 PMEVTYPER3_EL0, Performance Monitors Event Type Registers

Configures event counter 3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-144: AArch64_pmevtyper3_el0 bit assignments

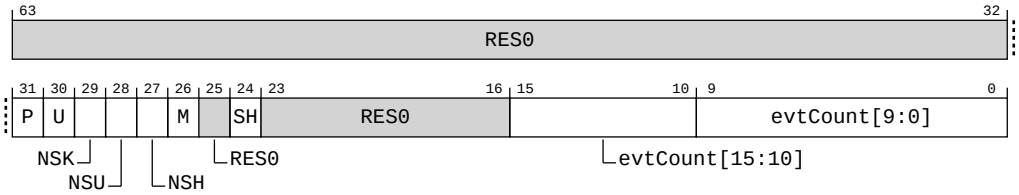


Table A-347: PMEVTYPER3_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter AArch64-PMEVCNTR<n>_EL0. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVTYPER3_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b011

MSR PMEVTYPER3_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b011

Accessibility

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.

- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVTYPER3_EL0

```
if 3 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[3];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[3];
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMEVTYPER_EL0[3];
```

MSR PMEVTYPER3_EL0, <Xt>

```
if 3 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[3] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[3] = X[t, 64];
```

A.6.15 PMEVTYPER4_EL0, Performance Monitors Event Type Registers

Configures event counter 4.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-145: AArch64_pmevtyper4_el0 bit assignments

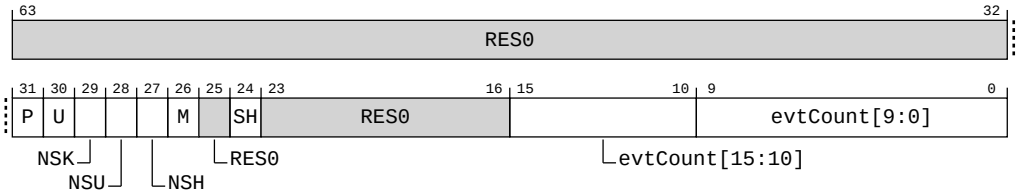


Table A-350: PMEVTYPER4_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter AArch64-PMEVCNTR<n>_EL0. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVTYPER4_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b100

MSR PMEVTYPER4_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b100

Accessibility

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.

- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVTYPER4_EL0

```
if 4 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 4 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[4];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[4];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[4];
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMEVTYPER_EL0[4];
```

MSR PMEVTYPER4_EL0, <Xt>

```
if 4 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 4 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[4] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[4] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[4] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[4] = X[t, 64];
```

A.6.16 PMEVTYPER5_EL0, Performance Monitors Event Type Registers

Configures event counter 5.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-146: AArch64_pmevtyper5_el0 bit assignments

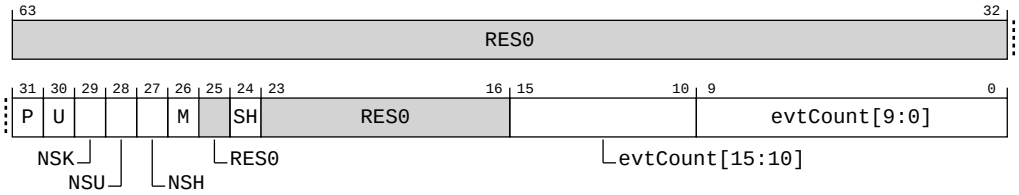


Table A-353: PMEVTYPER5_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If Secure EL2 is implemented, and EL3 is implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter AArch64-PMEVCNTR<n>_EL0. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are **CONSTRAINED UNPREDICTABLE**, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an **UNKNOWN** value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.

MRS <Xt>, PMEVTYPER5_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b101

MSR PMEVTYPER5_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1100	0b101

Accessibility

PMEVTYPER<n>_EL0 can also be accessed by using AArch64-PMXEVTYPER_EL0 with AArch64-PMSELR_EL0.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.

- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if <n> is an UNKNOWN value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.


Note

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_EL0.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, AArch64-MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see AArch64-MDCR_EL2.HPMN.
MRS <Xt>, PMEVTYPER5_EL0

```
if 5 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[5];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[5];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMEVTYPER_EL0[5];
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = PMEVTYPER_EL0[5];
```

MSR PMEVTYPER5_EL0, <Xt>

```
if 5 >= NUM_PMU_COUNTERS then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[5] = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[5] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMEVTYPER_EL0[5] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[5] = X[t, 64];
```

A.7 AArch64 GIC system registers summary

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

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

Table A-356: GIC system registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Interrupt Priority Mask Register
ICV_PMR_EL1	3	0	C4	C6	0	–	64-bit	Interrupt Controller Virtual Interrupt Priority Mask Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICC_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Interrupt Acknowledge Register 0
ICV_IAR0_EL1	3	0	C12	C8	0	–	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 0
ICC_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller End Of Interrupt Register 0
ICV_EOIR0_EL1	3	0	C12	C8	1	–	64-bit	Interrupt Controller Virtual End Of Interrupt Register 0
ICC_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 0
ICV_HPPIR0_EL1	3	0	C12	C8	2	–	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
ICV_BPR0_EL1	3	0	C12	C8	3	–	64-bit	Interrupt Controller Virtual Binary Point Register 0
ICC_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	0	C12	C8	4	–	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	0	C12	C9	0	–	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICC_DIR_EL1	3	0	C12	C11	1	–	64-bit	Interrupt Controller Deactivate Interrupt Register
ICV_DIR_EL1	3	0	C12	C11	1	–	64-bit	Interrupt Controller Deactivate Virtual Interrupt Register
ICC_RPR_EL1	3	0	C12	C11	3	–	64-bit	Interrupt Controller Running Priority Register
ICV_RPR_EL1	3	0	C12	C11	3	–	64-bit	Interrupt Controller Virtual Running Priority Register
ICC_SGI1R_EL1	3	0	C12	C11	5	–	64-bit	Interrupt Controller Software Generated Interrupt Group 1 Register
ICC_ASGI1R_EL1	3	0	C12	C11	6	–	64-bit	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
ICC_SGI0R_EL1	3	0	C12	C11	7	–	64-bit	Interrupt Controller Software Generated Interrupt Group 0 Register
ICC_IAR1_EL1	3	0	C12	C12	0	–	64-bit	Interrupt Controller Interrupt Acknowledge Register 1
ICV_IAR1_EL1	3	0	C12	C12	0	–	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 1
ICC_EOIR1_EL1	3	0	C12	C12	1	–	64-bit	Interrupt Controller End Of Interrupt Register 1
ICV_EOIR1_EL1	3	0	C12	C12	1	–	64-bit	Interrupt Controller Virtual End Of Interrupt Register 1
ICC_HPPIR1_EL1	3	0	C12	C12	2	–	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 1
ICV_HPPIR1_EL1	3	0	C12	C12	2	–	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
ICC_BPR1_EL1	3	0	C12	C12	3	–	64-bit	Interrupt Controller Binary Point Register 1
ICV_BPR1_EL1	3	0	C12	C12	3	–	64-bit	Interrupt Controller Virtual Binary Point Register 1
ICC_CTLR_EL1	3	0	C12	C12	4	–	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	0	C12	C12	4	–	64-bit	Interrupt Controller Virtual Control Register
ICC_SRE_EL1	3	0	C12	C12	5	–	64-bit	Interrupt Controller System Register Enable register (EL1)
ICC_IGRPEN0_EL1	3	0	C12	C12	6	–	64-bit	Interrupt Controller Interrupt Group 0 Enable register
ICV_IGRPEN0_EL1	3	0	C12	C12	6	–	64-bit	Interrupt Controller Virtual Interrupt Group 0 Enable register
ICC_IGRPEN1_EL1	3	0	C12	C12	7	–	64-bit	Interrupt Controller Interrupt Group 1 Enable register
ICV_IGRPEN1_EL1	3	0	C12	C12	7	–	64-bit	Interrupt Controller Virtual Interrupt Group 1 Enable register
ICH_AP0R0_EL2	3	4	C12	C8	0	–	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R0_EL2	3	4	C12	C9	0	–	64-bit	Interrupt Controller Hyp Active Priorities Group 1 Registers
ICC_SRE_EL2	3	4	C12	C9	5	–	64-bit	Interrupt Controller System Register Enable register (EL2)
ICH_HCR_EL2	3	4	C12	C11	0	–	64-bit	Interrupt Controller Hyp Control Register
ICH_VTR_EL2	3	4	C12	C11	1	–	64-bit	Interrupt Controller VGIC Type Register
ICH_MISR_EL2	3	4	C12	C11	2	–	64-bit	Interrupt Controller Maintenance Interrupt State Register
ICH_EISR_EL2	3	4	C12	C11	3	–	64-bit	Interrupt Controller End of Interrupt Status Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICH_ELRSR_EL2	3	4	C12	C11	5	–	64-bit	Interrupt Controller Empty List Register Status Register
ICH_VMCR_EL2	3	4	C12	C11	7	–	64-bit	Interrupt Controller Virtual Machine Control Register
ICH_LR0_EL2	3	4	C12	C12	0	–	64-bit	Interrupt Controller List Registers
ICH_LR1_EL2	3	4	C12	C12	1	–	64-bit	Interrupt Controller List Registers
ICH_LR2_EL2	3	4	C12	C12	2	–	64-bit	Interrupt Controller List Registers
ICH_LR3_EL2	3	4	C12	C12	3	–	64-bit	Interrupt Controller List Registers
ICC_CTLR_EL3	3	6	C12	C12	4	–	64-bit	Interrupt Controller Control Register (EL3)
ICC_SRE_EL3	3	6	C12	C12	5	–	64-bit	Interrupt Controller System Register Enable register (EL3)
ICC_IGRPEN1_EL3	3	6	C12	C12	7	–	64-bit	Interrupt Controller Interrupt Group 1 Enable register (EL3)

A.7.1 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers

Provides information about Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-147: AArch64_icc_ap0r0_el1 bit assignments

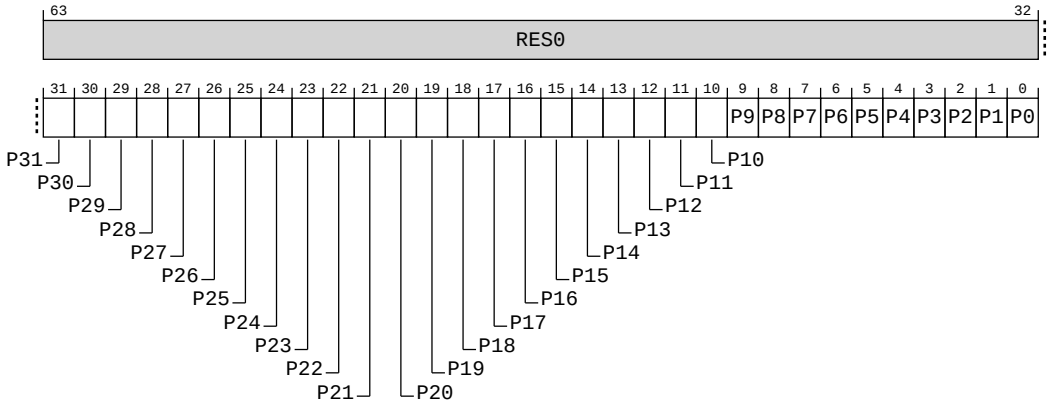


Table A-357: ICC_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	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].	x
[30]	P30	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].	x
[29]	P29	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].	x

Bits	Name	Description	Reset
[28]	P28	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].	x
[27]	P27	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].	x
[26]	P26	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].	x
[25]	P25	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].	x
[24]	P24	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].	x

Bits	Name	Description	Reset
[23]	P23	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].	x
[22]	P22	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].	x
[21]	P21	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].	x
[20]	P20	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].	x
[19]	P19	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].	x

Bits	Name	Description	Reset
[18]	P18	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].	x
[17]	P17	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].	x
[16]	P16	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].	x
[15]	P15	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].	x
[14]	P14	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].	x

Bits	Name	Description	Reset
[13]	P13	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].	x
[12]	P12	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].	x
[11]	P11	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].	x
[10]	P10	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].	x
[9]	P9	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].	x

Bits	Name	Description	Reset
[8]	P8	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].	x
[7]	P7	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].	x
[6]	P6	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].	x
[5]	P5	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].	x
[4]	P4	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].	x

Bits	Name	Description	Reset
[3]	P3	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].	x
[2]	P2	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].	x
[1]	P1	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].	x
[0]	P0	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].	x

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in **UNPREDICTABLE** behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP0R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP0R2_EL1 and ICC_AP0R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- ICC_AP0R<n>_EL1.
- Secure AArch64-ICC_AP1R<n>_EL1.
- Non-secure AArch64-ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b100

MSR ICC_AP0R0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b100

Accessibility

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP0R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP0R2_EL1 and ICC_AP0R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- ICC_AP0R<n>_EL1.

- Secure AArch64-ICC_AP1R<n>_EL1.
- Non-secure AArch64-ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_AP0R_EL1[0];
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
```

MSR ICC_AP0R0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_AP0R_EL1[0] = X[t, 64];
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
```

```
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
```

A.7.2 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers

Provides information about virtual Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-148: AArch64_icv_ap0r0_el1 bit assignments

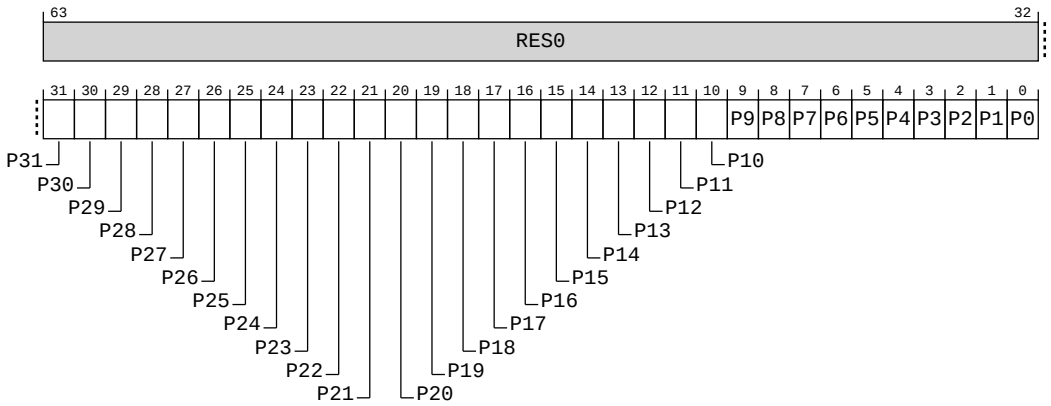


Table A-360: ICV_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[30]	P30	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[29]	P29	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[28]	P28	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[27]	P27	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[26]	P26	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[25]	P25	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[24]	P24	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[23]	P23	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[22]	P22	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[21]	P21	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[20]	P20	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[19]	P19	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[18]	P18	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[17]	P17	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[16]	P16	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[15]	P15	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[14]	P14	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[13]	P13	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[12]	P12	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[11]	P11	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[10]	P10	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[9]	P9	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[8]	P8	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[7]	P7	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[6]	P6	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[5]	P5	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[4]	P4	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[3]	P3	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[2]	P2	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[1]	P1	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[0]	P0	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

Writing to these registers with any value other than the last read value of the register (or `0x00000000` when there are no Group 0 active priorities) might result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP0R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV_AP0R2_EL1 and ICV_AP0R3_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are **UNDEFINED**.

Writing to the active priority registers in any order other than the following order might result in **UNPREDICTABLE** behavior of the interrupt prioritization system:

- ICV_AP0R<n>_EL1.
- AArch64-ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b100

MSR ICC_AP0R0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b100

Accessibility

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP0R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV_AP0R2_EL1 and ICV_AP0R3_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order might result in UNPREDICTABLE behavior of the interrupt prioritization system:

- ICV_AP0R<n>_EL1.
- AArch64-ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_AP0R_EL1[0];
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_AP0R_EL1[0];
```

MSR ICC_AP0R0_EL1, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_AP0R_EL1[0] = X[t, 64];
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_AP0R_EL1[0] = X[t, 64];
```

A.7.3 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers

Provides information about Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-149: AArch64_icc_ap1r0_el1 bit assignments

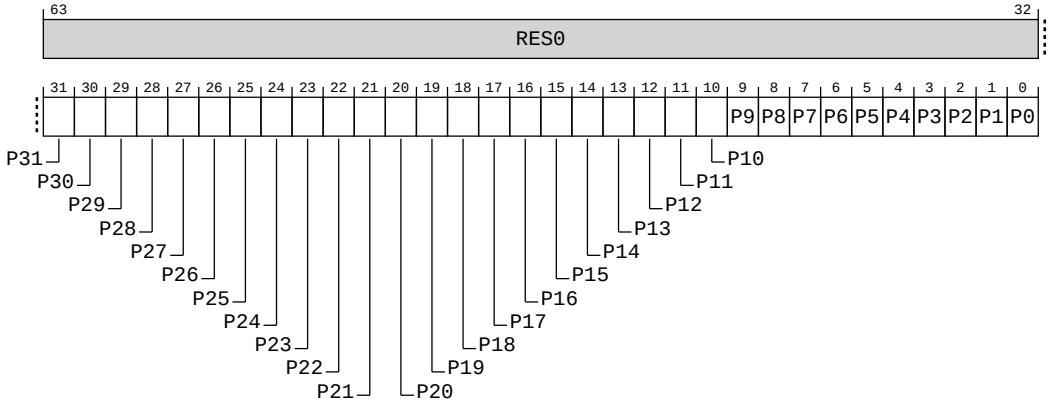


Table A-363: ICC_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	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.	x

Bits	Name	Description	Reset
[30]	P30	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.	x
[29]	P29	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.	x
[28]	P28	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.	x

Bits	Name	Description	Reset
[27]	P27	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.	x
[26]	P26	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.	x
[25]	P25	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.	x

Bits	Name	Description	Reset
[24]	P24	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.	x
[23]	P23	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.	x
[22]	P22	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.	x

Bits	Name	Description	Reset
[21]	P21	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.	x
[20]	P20	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.	x
[19]	P19	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.	x

Bits	Name	Description	Reset
[18]	P18	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.	x
[17]	P17	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.	x
[16]	P16	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.	x

Bits	Name	Description	Reset
[15]	P15	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.	x
[14]	P14	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.	x
[13]	P13	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.	x

Bits	Name	Description	Reset
[12]	P12	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.	x
[11]	P11	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.	x
[10]	P10	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.	x

Bits	Name	Description	Reset
[9]	P9	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.	x
[8]	P8	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.	x
[7]	P7	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.	x

Bits	Name	Description	Reset
[6]	P6	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.	x
[5]	P5	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.	x
[4]	P4	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.	x

Bits	Name	Description	Reset
[3]	P3	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.	x
[2]	P2	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.	x
[1]	P1	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.	x

Bits	Name	Description	Reset
[0]	P0	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.	x

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

Writing to these registers with any value other than the last read value of the register (or `0x00000000` when there are no Group 1 active priorities) might result in **UNPREDICTABLE** behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP1R2_EL1 and ICC_AP1R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- AArch64-ICC_AP0R<n>_EL1.
- Secure ICC_AP1R<n>_EL1.
- Non-secure ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b000

MSR ICC_AP1R0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b000

Accessibility

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC_AP1R2_EL1 and ICC_AP1R3_EL1 are only implemented in implementations that support 7 or more bits of priority. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- AArch64-ICC_AP0R<n>_EL1.
- Secure ICC_AP1R<n>_EL1.
- Non-secure ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_AP1R_EL1[0];
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
```

MSR ICC_AP1R0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_AP1R_EL1[0] = X[t, 64];
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
```

A.7.4 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers

Provides information about virtual Group 1 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-150: AArch64_icv_ap1r0_el1 bit assignments

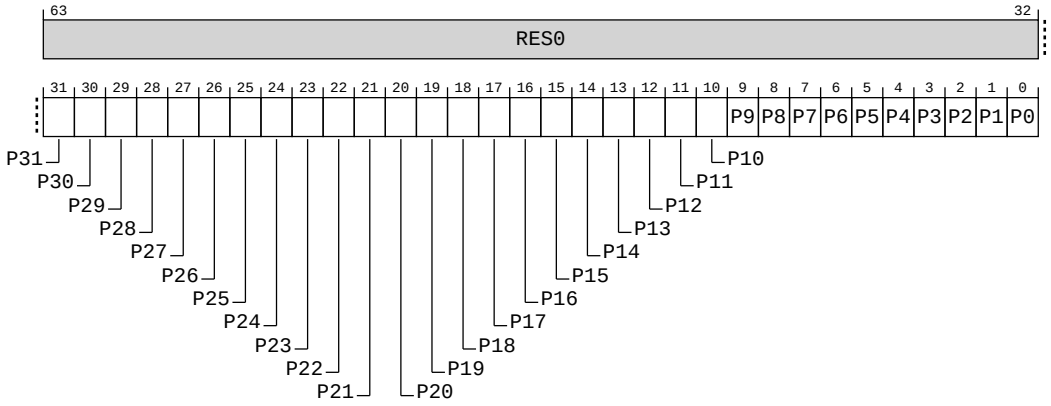


Table A-366: ICV_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31]	P31	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[30]	P30	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[29]	P29	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[28]	P28	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[27]	P27	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[26]	P26	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[25]	P25	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[24]	P24	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[23]	P23	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[22]	P22	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[21]	P21	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[20]	P20	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[19]	P19	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[18]	P18	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[17]	P17	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[16]	P16	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[15]	P15	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[14]	P14	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[13]	P13	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[12]	P12	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[11]	P11	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[10]	P10	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[9]	P9	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[8]	P8	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[7]	P7	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[6]	P6	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[5]	P5	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[4]	P4	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[3]	P3	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[2]	P2	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[1]	P1	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[0]	P0	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

Writing to these registers with any value other than the last read value of the register (or `0x00000000` when there are no Group 1 active priorities) might result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are **UNDEFINED**.

Writing to the active priority registers in any order other than the following order might result in **UNPREDICTABLE** behavior of the interrupt prioritization system:

- AArch64-ICV_AP0R<n>_EL1.
- ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b000

MSR ICC_AP1R0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b000

Accessibility

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order might result in UNPREDICTABLE behavior of the interrupt prioritization system:

- AArch64-ICV_AP0R<n>_EL1.
- ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_AP1R_EL1[0];
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
```

```
            X[t, 64] = ICC_AP1R_EL1_S[0];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[0];
```

MSR ICC_AP1R0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_AP1R_EL1[0] = X[t, 64];
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[0] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[0] = X[t, 64];
```

A.7.5 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-151: AArch64_icc_ctlr_el1 bit assignments

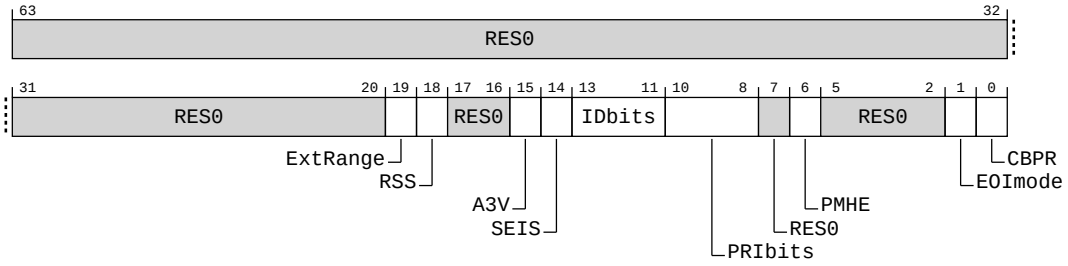


Table A-369: ICC_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x
[18]	RSS	Range Selector Support. Possible values are: **0b0** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	x
[17:16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **0b1** The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: **0b0** The CPU interface logic does not support local generation of SEIs.	x

Bits	Name	Description	Reset
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported: **0b000** 16 bits.	xxx
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of ext-GICD_CTLR.DS. For physical accesses, this field determines the minimum value of AArch64-ICC_BPR0_EL1. If EL3 is implemented, physical accesses return the value from AArch64-ICC_CTLR_EL3.PRIbits. **0b100** 5 bits of priority are implemented	xxx
[7]	**RES0**	Reserved	**RES0**
[6]	PMHE	Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution: **0b0** Disables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution. **0b1** Enables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution. If EL3 is implemented, this bit is an alias of AArch64-ICC_CTLR_EL3.PMHE. Whether this bit can be written as part of an access to this register depends on the value of ext-GICD_CTLR.DS: • If ext-GICD_CTLR.DS == 0, this bit is read-only. • If ext-GICD_CTLR.DS == 1, this bit is read/write.	x
[5:2]	**RES0**	Reserved	**RES0**
[1]	EOImode	EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt: **0b0** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. The Secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1S. The Non-secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1NS	x

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.	x

Access

MRS <Xt>, ICC_CTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

MSR ICC_CTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
```

A.7.6 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-152: AArch64_icv_ctlr_el1 bit assignments

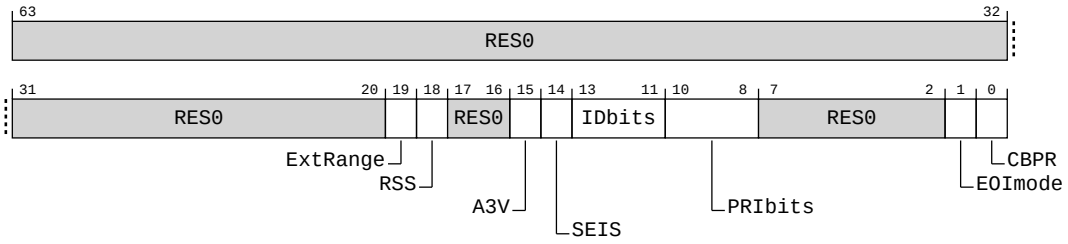


Table A-372: ICV_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x

Bits	Name	Description	Reset
[18]	RSS	Range Selector Support. Possible values are: **0b0** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	x
[17:16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **0b1** The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs: **0b0** The virtual CPU interface logic does not support local generation of SEIs.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported: **0b000** 16 bits.	xxx
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation must implement at least 32 levels of physical priority (5 priority bits). **Note:** This field always returns the number of priority bits implemented. The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPR0_EL1 and AArch64-ICV_BPR1_EL1. **0b100** 5 bits of priority are implemented	xxx
[7:2]	**RES0**	Reserved	**RES0**
[1]	EOImode	Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt: **0b0** AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICV_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.	x
[0]	CBPR	Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts: **0b0** AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts. **0b1** Non-secure reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Non-secure writes to AArch64-ICV_BPR1_EL1 are ignored. Secure reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1. Secure writes of AArch64-ICV_BPR1_EL1 modify AArch64-ICV_BPR0_EL1.	x

Access

MRS <Xt>, ICC_CTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

MSR ICC_CTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
```

MSR ICC_CTLR_EL1, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t, 64];
        else
            ICC_CTLR_EL1_NS = X[t, 64];
```

A.7.7 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Registers

Provides information about Group 0 virtual active priorities for EL2.

Configurations

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

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

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-153: AArch64_ich_ap0r0_el2 bit assignments

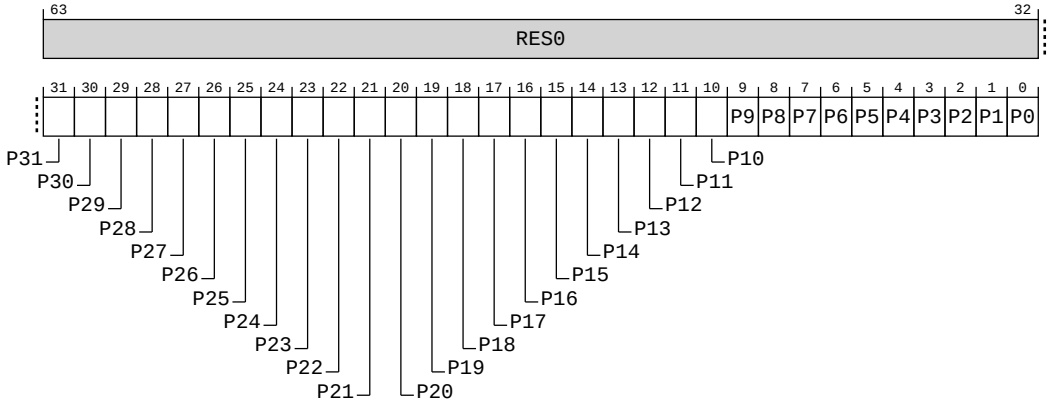


Table A-375: ICH_AP0R0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P31	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[30]	P30	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[29]	P29	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[28]	P28	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[27]	P27	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[26]	P26	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[25]	P25	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[24]	P24	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[23]	P23	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[22]	P22	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[21]	P21	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[20]	P20	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[19]	P19	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[18]	P18	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[17]	P17	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[16]	P16	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[15]	P15	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[14]	P14	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[13]	P13	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[12]	P12	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[11]	P11	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[10]	P10	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[9]	P9	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[8]	P8	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[7]	P7	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[6]	P6	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[5]	P5	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[4]	P4	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[3]	P3	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[2]	P2	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[1]	P1	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[0]	P0	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP0R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP0R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP0R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP0R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP0R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP0R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Software must ensure that ICH_AP0R<n>_EL2 is 0 for legacy VMs otherwise behavior is **UNPREDICTABLE**. For more information about support for legacy VMs, see 'Support for legacy operation of VMs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

The active priorities for Group 0 and Group 1 interrupts for legacy VMs are held in AArch64-ICH_AP1R<n>_EL2 and reads and writes to GICV_APR access AArch64-ICH_AP1R<n>_EL2. This means that ICH_AP0R<n>_EL2 is inaccessible to legacy VMs.

Access

ICH_AP0R1_EL2 is only implemented in implementations that support 6 or more bits of preemption. ICH_AP0R2_EL2 and ICH_AP0R3_EL2 are only implemented in implementations that support 7 bits of preemption. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits

Writing to these registers with any value other than the last read value of the register (or 0x00000000 for a newly set up virtual machine) can result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system allowing either:

- Virtual interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution at EL1 or EL0.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- ICH_AP0R<n>_EL2.
- AArch64-ICH_AP1R<n>_EL2.

Having the bit corresponding to a priority set in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 can result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.

MRS <Xt>, ICH_AP0R0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1000	0b000

MSR ICH_AP0R0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1000	0b000

Accessibility

ICH_AP0R1_EL2 is only implemented in implementations that support 6 or more bits of preemption. ICH_AP0R2_EL2 and ICH_AP0R3_EL2 are only implemented in implementations that support 7 bits of preemption. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits

Writing to these registers with any value other than the last read value of the register (or 0x00000000 for a newly set up virtual machine) can result in UNPREDICTABLE behavior of the virtual interrupt prioritization system allowing either:

- Virtual interrupts that should preempt execution to not preempt execution.

- Interrupts that should not preempt execution to preempt execution at EL1 or EL0.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- ICH_AP0R<n>_EL2.
- AArch64-ICH_AP1R<n>_EL2.

Having the bit corresponding to a priority set in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 can result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.
MRS <Xt>, ICH_AP0R0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_AP0R_EL2[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_AP0R_EL2[0];
```

MSR ICH_AP0R0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_AP0R_EL2[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_AP0R_EL2[0] = X[t, 64];
```

A.7.8 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Registers

Provides information about Group 1 virtual active priorities for EL2.

Configurations

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

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

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-154: AArch64_ich_ap1r0_el2 bit assignments

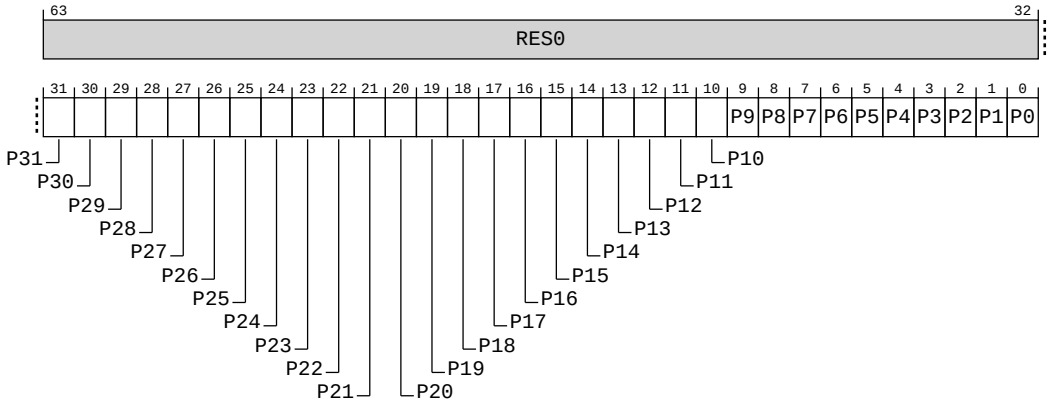


Table A-378: ICH_AP1R0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P31	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[30]	P30	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[29]	P29	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[28]	P28	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[27]	P27	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[26]	P26	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[25]	P25	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[24]	P24	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[23]	P23	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[22]	P22	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[21]	P21	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[20]	P20	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[19]	P19	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[18]	P18	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[17]	P17	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[16]	P16	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[15]	P15	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[14]	P14	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[13]	P13	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[12]	P12	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[11]	P11	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[10]	P10	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[9]	P9	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[8]	P8	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[7]	P7	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[6]	P6	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[5]	P5	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[4]	P4	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[3]	P3	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[2]	P2	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[1]	P1	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

Bits	Name	Description	Reset
[0]	P0	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. The correspondence between priority levels and bits depends on the number of bits of priority that are implemented. If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3]. If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and: • The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2]. • The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2]. If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and: • The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1]. • The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1]. • The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1]. • The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1]. **Note:** Having the bit corresponding to a priority set to 1 in both AArch64-ICH_AP0R<n>_EL2 and ICH_AP1R<n>_EL2 might result in **UNPREDICTABLE** behavior of the interrupt prioritization system for virtual interrupts.	0b0

This register is always used for legacy VMs, regardless of the group of the virtual interrupt. Reads and writes to ext-GICV_APR<n> access AArch64-ICH_AP1R<n>_EL2. For more information about support for legacy VMs, see 'Support for legacy operation of VMs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Access

ICH_AP1R1_EL2 is only implemented in implementations that support 6 or more bits of preemption. ICH_AP1R2_EL2 and ICH_AP1R3_EL2 are only implemented in implementations that support 7 bits of preemption. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits

Writing to these registers with any value other than the last read value of the register (or 0x00000000 for a newly set up virtual machine) can result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system allowing either:

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

MRS <Xt>, ICH_AP1R0_EL2

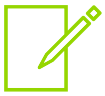
op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b000

MSR ICH_AP1R0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b000

Accessibility

ICH_AP1R1_EL2 is only implemented in implementations that support 6 or more bits of preemption. ICH_AP1R2_EL2 and ICH_AP1R3_EL2 are only implemented in implementations that support 7 bits of preemption. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by AArch64-ICH_VTR_EL2.PREbits

Writing to these registers with any value other than the last read value of the register (or 0x00000000 for a newly set up virtual machine) can result in UNPREDICTABLE behavior of the virtual interrupt prioritization system allowing either:

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:
MRS <Xt>, ICH_AP1R0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_AP1R_EL2[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_AP1R_EL2[0];
```

MSR ICH_AP1R0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_AP1R_EL2[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_AP1R_EL2[0] = X[t, 64];
```

A.7.9 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualization features.

Configurations

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

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

Attributes

Width

64

Functional group

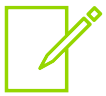
GIC system registers

Access type

See bit descriptions

Reset value

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



Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-155: AArch64_ich_vtr_el2 bit assignments

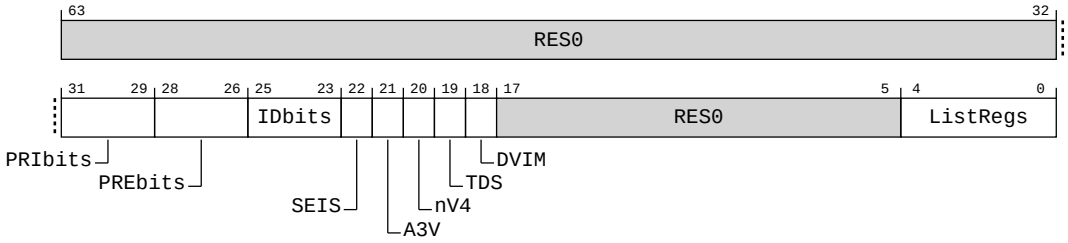


Table A-381: ICH_VTR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:29]	PRIbits	Priority bits. The number of virtual priority bits implemented, minus one. An implementation must implement at least 32 levels of virtual priority (5 priority bits). This field is an alias of AArch64-ICV_CTLR_EL1.PRIbits. **0b100** 5 virtual priority bits are implemented	xxx
[28:26]	PREbits	The number of virtual preemption bits implemented, minus one. An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits). The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits. The maximum value of this field is 6, indicating 7 bits of preemption. This field determines the minimum value of AArch64-ICH_VMCR_EL2.VBPR0. **0b100** 5 virtual pre-emption bits are implemented	xxx
[25:23]	IDbits	The number of virtual interrupt identifier bits supported: **0b000** 16 bits.	xxx
[22]	SEIS	SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs: **0b0** The virtual CPU interface logic does not support generation of SEIs.	x
[21]	A3V	Affinity 3 Valid. Possible values are: **0b1** The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	x
[20]	nV4	Direct injection of virtual interrupts not supported. Possible values are: **0b0** The CPU interface logic supports direct injection of virtual interrupts.	x

Bits	Name	Description	Reset
[19]	TDS	Separate trapping of EL1 writes to AArch64-ICV_DIR_EL1 supported. **0b1** Implementation supports AArch64-ICH_HCR_EL2.TDIR.	x
[18]	DVIM	Masking of directly-injected virtual interrupts. **0b0** Masking of Directly-injected Virtual Interrupts not supported.	x
[17:5]	**RES0**	Reserved	**RES0**
[4:0]	ListRegs	The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented. **0b00011** 4 List registers	5{x}

Access

MRS <Xt>, ICH_VTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b001

Accessibility

MRS <Xt>, ICH_VTR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_VTR_EL2;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_VTR_EL2;
```

A.7.10 ICH_LR0_EL2, Interrupt Controller List Registers

Provides interrupt context information for the virtual CPU interface.

Configurations

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

If list register n is not implemented, then accesses to this register are UNDEFINED.

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

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-156: AArch64_ich_lr0_el2 bit assignments

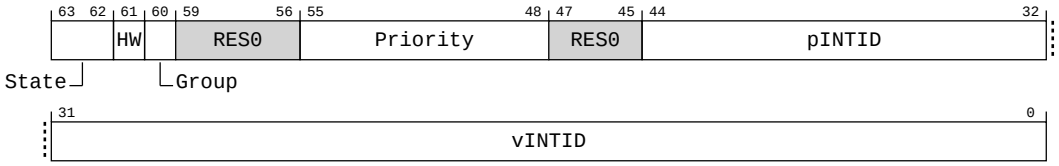


Table A-383: ICH_LR0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active. The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the invalid state are ignored, except for the purpose of generating virtual maintenance interrupts. For hardware interrupts, the pending and active state is held in the physical Distributor rather than the virtual CPU interface. A hypervisor must only use the pending and active state for software originated interrupts, which are typically associated with virtual devices, or SGIs.	xx

Bits	Name	Description	Reset
[61]	HW	Indicates whether this virtual interrupt maps directly to a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the pINTID field indicates. **0b0** The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated. **0b1** The interrupt maps directly to a hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using the pINTID field from this register to indicate the physical interrupt ID. If AArch64-ICH_VMCR_EL2.VEOIM is 0, this request corresponds to a write to AArch64-ICC_EOIR0_EL1 or AArch64-ICC_EOIR1_EL1. Otherwise, it corresponds to a write to AArch64-ICC_DIR_EL1.	x
[60]	Group	Indicates the group for this virtual interrupt. **0b0** This is a Group 0 virtual interrupt. AArch64-ICH_VMCR_EL2.VFIQEn determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and AArch64-ICH_VMCR_EL2.VENG0 enables signaling of this interrupt to the virtual machine. **0b1** This is a Group 1 virtual interrupt, signaled as a virtual IRQ. AArch64-ICH_VMCR_EL2.VENG1 enables the signalling of this interrupt to the virtual machine. If AArch64-ICH_VMCR_EL2.VCBPR is 0, then AArch64-ICC_BPR1_EL1 determines if a pending Group 1 interrupt has sufficient priority to preempt current execution. Otherwise, AArch64-ICH_LR<n>_EL2 determines preemption.	x
[59:56]	**RES0**	Reserved	**RES0**
[55:48]	Priority	The priority of this interrupt. It is IMPLEMENTATION DEFINED how many bits of priority are implemented, though at least five bits must be implemented. Unimplemented bits are **RES0** and start from bit[48] up to bit[50]. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.PRIbits.	8{x}
[47:45]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[44:32]	pINTID	Physical INTID, for hardware interrupts. When ICH_LR<n>_EL2.HW is 0 (there is no corresponding physical interrupt), this field has the following meaning: • Bits[44:42] : **RES0**. • Bit[41] : EOI. If this bit is 1, then when the interrupt identified by vINTID is deactivated, a maintenance interrupt is asserted. • Bits[40:32] : **RES0**. When ICH_LR<n>_EL2.HW is 1 (there is a corresponding physical interrupt): • This field indicates the physical INTID. This field is only required to implement enough bits to hold a valid value for the implemented INTID size. Any unused higher order bits are **RES0**. • When AArch64-ICC_CTLR_EL1.ExtRange is 0, then bits[44:42] of this field are **RES0**. • If the value of pINTID is not a vald INTID, behavior is UNPREDICTABLE. If the value of pINTID indicates a PPI, this field applies to the PPI associated with this same physical PE ID as the virtual CPU interface requesting the deactivation. A hardware physical identifier is only required in List Registers for interrupts that require deactivation. This means only 13 bits of Physical INTID are required, regardless of the number specified by AArch64-ICC_CTLR_EL1.IDbits.	13{x}
[31:0]	vINTID	Virtual INTID of the interrupt. If the value of vINTID is 1020-1023 and ICH_LR<n>_EL2.State!=0b00 (Inactive), behavior is UNPREDICTABLE. Behavior is UNPREDICTABLE if two or more List Registers specify the same vINTID when: • ICH_LR<n>_EL2.State == 0b01. • ICH_LR<n>_EL2.State == 0b10. • ICH_LR<n>_EL2.State == 0b11. It is IMPLEMENTATION DEFINED how many bits are implemented, though at least 16 bits must be implemented. Unimplemented bits are **RES0**. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.IDbits. When AArch64-ICC_SRE_EL1.SRE == 0, specifying a vINTID in the LPI range is UNPREDICTABLE **Note:** When a VM is using memory-mapped access to the GIC, software must ensure that the correct source PE ID is provided in bits[12:10].	32{x}

Access

MRS <Xt>, ICH_LR0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b000

MSR ICH_LR0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b000

Accessibility

MRS <Xt>, ICH_LR0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[0];
```

MSR ICH_LR0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_LR_EL2[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_LR_EL2[0] = X[t, 64];
```

A.7.11 ICH_LR1_EL2, Interrupt Controller List Registers

Provides interrupt context information for the virtual CPU interface.

Configurations

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

If list register n is not implemented, then accesses to this register are UNDEFINED.

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

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-157: AArch64_ich_lr1_el2 bit assignments

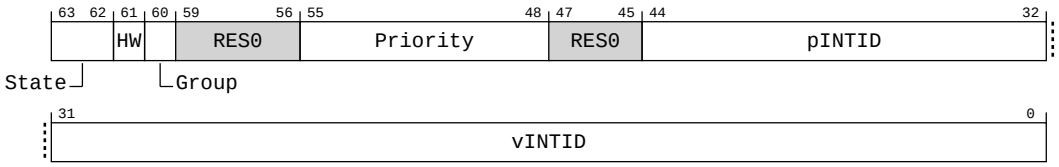


Table A-386: ICH_LR1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active. The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the invalid state are ignored, except for the purpose of generating virtual maintenance interrupts. For hardware interrupts, the pending and active state is held in the physical Distributor rather than the virtual CPU interface. A hypervisor must only use the pending and active state for software originated interrupts, which are typically associated with virtual devices, or SGIs.	xx

Bits	Name	Description	Reset
[61]	HW	Indicates whether this virtual interrupt maps directly to a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the pINTID field indicates. **0b0** The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated. **0b1** The interrupt maps directly to a hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using the pINTID field from this register to indicate the physical interrupt ID. If AArch64-ICH_VMCR_EL2.VEOIM is 0, this request corresponds to a write to AArch64-ICC_EOIR0_EL1 or AArch64-ICC_EOIR1_EL1. Otherwise, it corresponds to a write to AArch64-ICC_DIR_EL1.	x
[60]	Group	Indicates the group for this virtual interrupt. **0b0** This is a Group 0 virtual interrupt. AArch64-ICH_VMCR_EL2.VFIQEn determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and AArch64-ICH_VMCR_EL2.VENG0 enables signaling of this interrupt to the virtual machine. **0b1** This is a Group 1 virtual interrupt, signaled as a virtual IRQ. AArch64-ICH_VMCR_EL2.VENG1 enables the signalling of this interrupt to the virtual machine. If AArch64-ICH_VMCR_EL2.VCBPR is 0, then AArch64-ICC_BPR1_EL1 determines if a pending Group 1 interrupt has sufficient priority to preempt current execution. Otherwise, AArch64-ICH_LR<n>_EL2 determines preemption.	x
[59:56]	**RES0**	Reserved	**RES0**
[55:48]	Priority	The priority of this interrupt. It is IMPLEMENTATION DEFINED how many bits of priority are implemented, though at least five bits must be implemented. Unimplemented bits are **RES0** and start from bit[48] up to bit[50]. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.PRIbits.	8{x}
[47:45]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[44:32]	pINTID	Physical INTID, for hardware interrupts. When ICH_LR<n>_EL2.HW is 0 (there is no corresponding physical interrupt), this field has the following meaning: • Bits[44:42] : **RES0**. • Bit[41] : EOI. If this bit is 1, then when the interrupt identified by vINTID is deactivated, a maintenance interrupt is asserted. • Bits[40:32] : **RES0**. When ICH_LR<n>_EL2.HW is 1 (there is a corresponding physical interrupt): • This field indicates the physical INTID. This field is only required to implement enough bits to hold a valid value for the implemented INTID size. Any unused higher order bits are **RES0**. • When AArch64-ICC_CTLR_EL1.ExtRange is 0, then bits[44:42] of this field are **RES0**. • If the value of pINTID is not a vald INTID, behavior is UNPREDICTABLE. If the value of pINTID indicates a PPI, this field applies to the PPI associated with this same physical PE ID as the virtual CPU interface requesting the deactivation. A hardware physical identifier is only required in List Registers for interrupts that require deactivation. This means only 13 bits of Physical INTID are required, regardless of the number specified by AArch64-ICC_CTLR_EL1.IDbits.	13{x}
[31:0]	vINTID	Virtual INTID of the interrupt. If the value of vINTID is 1020-1023 and ICH_LR<n>_EL2.State!=0b00 (Inactive), behavior is UNPREDICTABLE. Behavior is UNPREDICTABLE if two or more List Registers specify the same vINTID when: • ICH_LR<n>_EL2.State == 0b01. • ICH_LR<n>_EL2.State == 0b10. • ICH_LR<n>_EL2.State == 0b11. It is IMPLEMENTATION DEFINED how many bits are implemented, though at least 16 bits must be implemented. Unimplemented bits are **RES0**. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.IDbits. When AArch64-ICC_SRE_EL1.SRE == 0, specifying a vINTID in the LPI range is UNPREDICTABLE **Note:** When a VM is using memory-mapped access to the GIC, software must ensure that the correct source PE ID is provided in bits[12:10].	32{x}

Access

MRS <Xt>, ICH_LR1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b001

MSR ICH_LR1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b001

Accessibility

MRS <Xt>, ICH_LR1_EL2

```
if 1 >= NUM_GIC_LIST_REGS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[1];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[1];
```

MSR ICH_LR1_EL2, <Xt>

```
if 1 >= NUM_GIC_LIST_REGS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_LR_EL2[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_LR_EL2[1] = X[t, 64];
```

A.7.12 ICH_LR2_EL2, Interrupt Controller List Registers

Provides interrupt context information for the virtual CPU interface.

Configurations

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

If list register n is not implemented, then accesses to this register are UNDEFINED.

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

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-158: AArch64_ich_lr2_el2 bit assignments

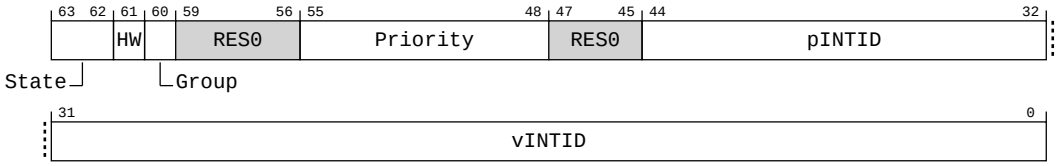


Table A-389: ICH_LR2_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active. The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the invalid state are ignored, except for the purpose of generating virtual maintenance interrupts. For hardware interrupts, the pending and active state is held in the physical Distributor rather than the virtual CPU interface. A hypervisor must only use the pending and active state for software originated interrupts, which are typically associated with virtual devices, or SGIs.	xx

Bits	Name	Description	Reset
[61]	HW	Indicates whether this virtual interrupt maps directly to a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the pINTID field indicates. **0b0** The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated. **0b1** The interrupt maps directly to a hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using the pINTID field from this register to indicate the physical interrupt ID. If AArch64-ICH_VMCR_EL2.VEOIM is 0, this request corresponds to a write to AArch64-ICC_EOIR0_EL1 or AArch64-ICC_EOIR1_EL1. Otherwise, it corresponds to a write to AArch64-ICC_DIR_EL1.	x
[60]	Group	Indicates the group for this virtual interrupt. **0b0** This is a Group 0 virtual interrupt. AArch64-ICH_VMCR_EL2.VFIQEn determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and AArch64-ICH_VMCR_EL2.VENG0 enables signaling of this interrupt to the virtual machine. **0b1** This is a Group 1 virtual interrupt, signaled as a virtual IRQ. AArch64-ICH_VMCR_EL2.VENG1 enables the signalling of this interrupt to the virtual machine. If AArch64-ICH_VMCR_EL2.VCBPR is 0, then AArch64-ICC_BPR1_EL1 determines if a pending Group 1 interrupt has sufficient priority to preempt current execution. Otherwise, AArch64-ICH_LR<n>_EL2 determines preemption.	x
[59:56]	**RES0**	Reserved	**RES0**
[55:48]	Priority	The priority of this interrupt. It is IMPLEMENTATION DEFINED how many bits of priority are implemented, though at least five bits must be implemented. Unimplemented bits are **RES0** and start from bit[48] up to bit[50]. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.PRIbits.	8{x}
[47:45]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[44:32]	pINTID	Physical INTID, for hardware interrupts. When ICH_LR<n>_EL2.HW is 0 (there is no corresponding physical interrupt), this field has the following meaning: • Bits[44:42] : **RES0**. • Bit[41] : EOI. If this bit is 1, then when the interrupt identified by vINTID is deactivated, a maintenance interrupt is asserted. • Bits[40:32] : **RES0**. When ICH_LR<n>_EL2.HW is 1 (there is a corresponding physical interrupt): • This field indicates the physical INTID. This field is only required to implement enough bits to hold a valid value for the implemented INTID size. Any unused higher order bits are **RES0**. • When AArch64-ICC_CTLR_EL1.ExtRange is 0, then bits[44:42] of this field are **RES0**. • If the value of pINTID is not a vald INTID, behavior is UNPREDICTABLE. If the value of pINTID indicates a PPI, this field applies to the PPI associated with this same physical PE ID as the virtual CPU interface requesting the deactivation. A hardware physical identifier is only required in List Registers for interrupts that require deactivation. This means only 13 bits of Physical INTID are required, regardless of the number specified by AArch64-ICC_CTLR_EL1.IDbits.	13{x}
[31:0]	vINTID	Virtual INTID of the interrupt. If the value of vINTID is 1020-1023 and ICH_LR<n>_EL2.State!=0b00 (Inactive), behavior is UNPREDICTABLE. Behavior is UNPREDICTABLE if two or more List Registers specify the same vINTID when: • ICH_LR<n>_EL2.State == 0b01. • ICH_LR<n>_EL2.State == 0b10. • ICH_LR<n>_EL2.State == 0b11. It is IMPLEMENTATION DEFINED how many bits are implemented, though at least 16 bits must be implemented. Unimplemented bits are **RES0**. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.IDbits. When AArch64-ICC_SRE_EL1.SRE == 0, specifying a vINTID in the LPI range is UNPREDICTABLE **Note:** When a VM is using memory-mapped access to the GIC, software must ensure that the correct source PE ID is provided in bits[12:10].	32{x}

Access

MRS <Xt>, ICH_LR2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b010

MSR ICH_LR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b010

Accessibility

MRS <Xt>, ICH_LR2_EL2

```
if 2 >= NUM_GIC_LIST_REGS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[2];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[2];
```

MSR ICH_LR2_EL2, <Xt>

```
if 2 >= NUM_GIC_LIST_REGS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_LR_EL2[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_LR_EL2[2] = X[t, 64];
```

A.7.13 ICH_LR3_EL2, Interrupt Controller List Registers

Provides interrupt context information for the virtual CPU interface.

Configurations

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

If list register n is not implemented, then accesses to this register are UNDEFINED.

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

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-159: AArch64_ich_lr3_el2 bit assignments

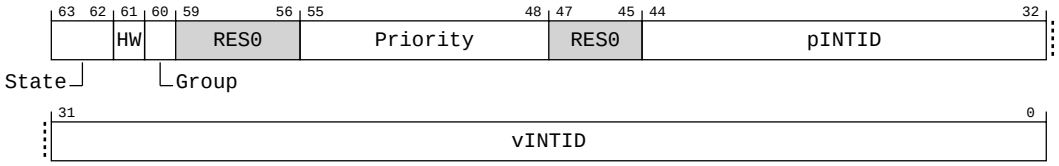


Table A-392: ICH_LR3_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active. The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the invalid state are ignored, except for the purpose of generating virtual maintenance interrupts. For hardware interrupts, the pending and active state is held in the physical Distributor rather than the virtual CPU interface. A hypervisor must only use the pending and active state for software originated interrupts, which are typically associated with virtual devices, or SGIs.	xx

Bits	Name	Description	Reset
[61]	HW	Indicates whether this virtual interrupt maps directly to a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the pINTID field indicates. **0b0** The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated. **0b1** The interrupt maps directly to a hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using the pINTID field from this register to indicate the physical interrupt ID. If AArch64-ICH_VMCR_EL2.VEOIM is 0, this request corresponds to a write to AArch64-ICC_EOIR0_EL1 or AArch64-ICC_EOIR1_EL1. Otherwise, it corresponds to a write to AArch64-ICC_DIR_EL1.	x
[60]	Group	Indicates the group for this virtual interrupt. **0b0** This is a Group 0 virtual interrupt. AArch64-ICH_VMCR_EL2.VFIQEn determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and AArch64-ICH_VMCR_EL2.VENG0 enables signaling of this interrupt to the virtual machine. **0b1** This is a Group 1 virtual interrupt, signaled as a virtual IRQ. AArch64-ICH_VMCR_EL2.VENG1 enables the signalling of this interrupt to the virtual machine. If AArch64-ICH_VMCR_EL2.VCBPR is 0, then AArch64-ICC_BPR1_EL1 determines if a pending Group 1 interrupt has sufficient priority to preempt current execution. Otherwise, AArch64-ICH_LR<n>_EL2 determines preemption.	x
[59:56]	**RES0**	Reserved	**RES0**
[55:48]	Priority	The priority of this interrupt. It is IMPLEMENTATION DEFINED how many bits of priority are implemented, though at least five bits must be implemented. Unimplemented bits are **RES0** and start from bit[48] up to bit[50]. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.PRIbits.	8{x}
[47:45]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[44:32]	pINTID	Physical INTID, for hardware interrupts. When ICH_LR<n>_EL2.HW is 0 (there is no corresponding physical interrupt), this field has the following meaning: • Bits[44:42] : **RES0**. • Bit[41] : EOI. If this bit is 1, then when the interrupt identified by vINTID is deactivated, a maintenance interrupt is asserted. • Bits[40:32] : **RES0**. When ICH_LR<n>_EL2.HW is 1 (there is a corresponding physical interrupt): • This field indicates the physical INTID. This field is only required to implement enough bits to hold a valid value for the implemented INTID size. Any unused higher order bits are **RES0**. • When AArch64-ICC_CTLR_EL1.ExtRange is 0, then bits[44:42] of this field are **RES0**. • If the value of pINTID is not a vald INTID, behavior is UNPREDICTABLE. If the value of pINTID indicates a PPI, this field applies to the PPI associated with this same physical PE ID as the virtual CPU interface requesting the deactivation. A hardware physical identifier is only required in List Registers for interrupts that require deactivation. This means only 13 bits of Physical INTID are required, regardless of the number specified by AArch64-ICC_CTLR_EL1.IDbits.	13{x}
[31:0]	vINTID	Virtual INTID of the interrupt. If the value of vINTID is 1020-1023 and ICH_LR<n>_EL2.State!=0b00 (Inactive), behavior is UNPREDICTABLE. Behavior is UNPREDICTABLE if two or more List Registers specify the same vINTID when: • ICH_LR<n>_EL2.State == 0b01. • ICH_LR<n>_EL2.State == 0b10. • ICH_LR<n>_EL2.State == 0b11. It is IMPLEMENTATION DEFINED how many bits are implemented, though at least 16 bits must be implemented. Unimplemented bits are **RES0**. The number of implemented bits can be discovered from AArch64-ICH_VTR_EL2.IDbits. When AArch64-ICC_SRE_EL1.SRE == 0, specifying a vINTID in the LPI range is UNPREDICTABLE **Note:** When a VM is using memory-mapped access to the GIC, software must ensure that the correct source PE ID is provided in bits[12:10].	32{x}

Access

MRS <Xt>, ICH_LR3_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b011

MSR ICH_LR3_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1100	0b011

Accessibility

MRS <Xt>, ICH_LR3_EL2

```
if 3 >= NUM_GIC_LIST_REGS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[3];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_LR_EL2[3];
```

MSR ICH_LR3_EL2, <Xt>

```
if 3 >= NUM_GIC_LIST_REGS then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_LR_EL2[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_LR_EL2[3] = X[t, 64];
```

A.7.14 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-160: AArch64_icc_ctlr_el3 bit assignments

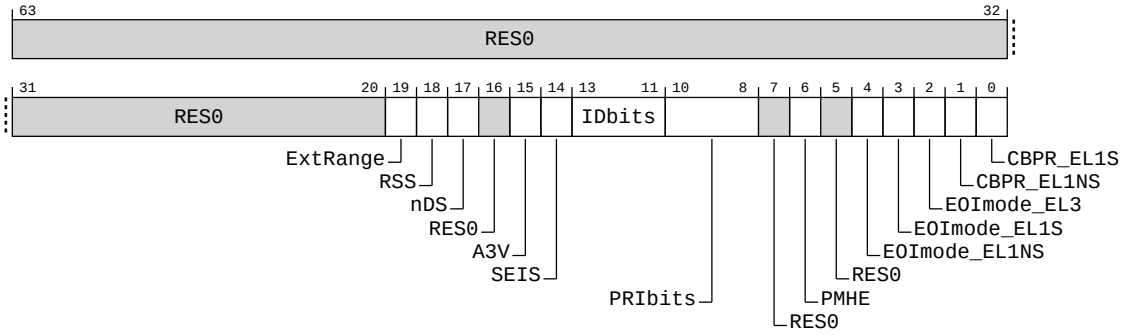


Table A-395: ICC_CTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x
[18]	RSS	Range Selector Support. **0b0** Targeted SGIs with affinity level 0 values of 0-15 are supported.	x
[17]	nDS	Disable Security not supported. Read-only and writes are ignored. **0b1** The CPU interface logic does not support disabling of security, and requires that security is not disabled.	x
[16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. **0b1** The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs: **0b0** The CPU interface logic does not support generation of SEIs.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported. **0b000** 16 bits.	xxx
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of ext-GICD_CTLR.DS. The division between group priority and subpriority is defined in the binary point registers AArch64-ICC_BPR0_EL1 and AArch64-ICC_BPR1_EL1. This field determines the minimum value of ICC_BPR0_EL1. **0b100** 5 bits of priority are implemented	xxx
[7]	**RES0**	Reserved	**RES0**
[6]	PMHE	Priority Mask Hint Enable. **0b0** Disables use of the priority mask register as a hint for interrupt distribution. **0b1** Enables use of the priority mask register as a hint for interrupt distribution. Software must write AArch64-ICC_PMR_EL1 to 0xFF before clearing this field to 0. • An implementation might choose to make this field **RAO/WI** if priority-based routing is always used • An implementation might choose to make this field RAZ/WI if priority-based routing is never used If EL3 is present, AArch64-ICC_CTLR_EL1.PMHE is an alias of ICC_CTLR_EL3.PMHE.	0b0
[5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4]	EOImode_EL1NS	EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0b0** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. If EL3 is present, AArch64-ICC_CTLR_EL1(NS).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS.	x
[3]	EOImode_EL1S	EOI mode for interrupts handled at Secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0b0** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. If EL3 is present, AArch64-ICC_CTLR_EL1(S).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.	x
[2]	EOImode_EL3	EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0b0** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.	x
[1]	CBPR_EL1NS	Common Binary Point Register, EL1 Non-secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2. **0b0** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts. **0b1** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to ext-GICC_BPR and AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPR0_EL1. If EL3 is present, AArch64-ICC_CTLR_EL1(NS).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1NS.	x

Bits	Name	Description	Reset
[0]	CBPR_EL1S	Common Binary Point Register, EL1 Secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupts at EL1 and EL2. **0b0** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts. **0b1** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPR0_EL1. If EL3 is present, AArch64-ICC_CTLR_EL1(S).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1S.	x

Access

MRS <Xt>, ICC_CTLR_EL3

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

MSR ICC_CTLR_EL3, <Xt>

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

Accessibility

MRS <Xt>, ICC_CTLR_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICC_CTLR_EL3;
```

MSR ICC_CTLR_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICC_CTLR_EL3 = X[t, 64];
```

A.8 AArch64 Generic Timer registers summary

The summary table provides an overview of all Generic Timer registers in the core. For more information about a register, click the register name in the table.

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

Table A-398: Generic Timer registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CNTKCTL_EL1	3	0	C14	C1	0	—	64-bit	Counter-timer Kernel Control register
CNTFRQ_EL0	3	3	C14	C0	0	—	64-bit	Counter-timer Frequency register
CNTPCT_EL0	3	3	C14	C0	1	—	64-bit	Counter-timer Physical Count register
CNTVCT_EL0	3	3	C14	C0	2	—	64-bit	Counter-timer Virtual Count register
CNTP_TVAL_EL0	3	3	C14	C2	0	—	64-bit	Counter-timer Physical Timer TimerValue register
CNTP_CTL_EL0	3	3	C14	C2	1	—	64-bit	Counter-timer Physical Timer Control register
CNTP_CVAL_EL0	3	3	C14	C2	2	—	64-bit	Counter-timer Physical Timer CompareValue register
CNTV_TVAL_EL0	3	3	C14	C3	0	—	64-bit	Counter-timer Virtual Timer TimerValue register
CNTV_CTL_EL0	3	3	C14	C3	1	—	64-bit	Counter-timer Virtual Timer Control register
CNTV_CVAL_EL0	3	3	C14	C3	2	—	64-bit	Counter-timer Virtual Timer CompareValue register
CNTVOFF_EL2	3	4	C14	C0	3	—	64-bit	Counter-timer Virtual Offset register
CNTHCTL_EL2	3	4	C14	C1	0	—	64-bit	Counter-timer Hypervisor Control register
CNTHP_TVAL_EL2	3	4	C14	C2	0	—	64-bit	Counter-timer Physical Timer TimerValue register (EL2)
CNTHP_CTL_EL2	3	4	C14	C2	1	—	64-bit	Counter-timer Hypervisor Physical Timer Control register
CNTHP_CVAL_EL2	3	4	C14	C2	2	—	64-bit	Counter-timer Physical Timer CompareValue register (EL2)
CNTHV_TVAL_EL2	3	4	C14	C3	0	—	64-bit	Counter-timer Virtual Timer TimerValue Register (EL2)
CNTHV_CTL_EL2	3	4	C14	C3	1	—	64-bit	Counter-timer Virtual Timer Control register (EL2)
CNTHV_CVAL_EL2	3	4	C14	C3	2	—	64-bit	Counter-timer Virtual Timer CompareValue register (EL2)
CNTHVS_TVAL_EL2	3	4	C14	C4	0	—	64-bit	Counter-timer Secure Virtual Timer TimerValue register (EL2)
CNTHVS_CTL_EL2	3	4	C14	C4	1	—	64-bit	Counter-timer Secure Virtual Timer Control register (EL2)
CNTHVS_CVAL_EL2	3	4	C14	C4	2	—	64-bit	Counter-timer Secure Virtual Timer CompareValue register (EL2)
CNTHPS_TVAL_EL2	3	4	C14	C5	0	—	64-bit	Counter-timer Secure Physical Timer TimerValue register (EL2)
CNTHPS_CTL_EL2	3	4	C14	C5	1	—	64-bit	Counter-timer Secure Physical Timer Control register (EL2)
CNTHPS_CVAL_EL2	3	4	C14	C5	2	—	64-bit	Counter-timer Secure Physical Timer CompareValue register (EL2)
CNTPS_TVAL_EL1	3	7	C14	C2	0	—	64-bit	Counter-timer Physical Secure Timer TimerValue register
CNTPS_CTL_EL1	3	7	C14	C2	1	—	64-bit	Counter-timer Physical Secure Timer Control register
CNTPS_CVAL_EL1	3	7	C14	C2	2	—	64-bit	Counter-timer Physical Secure Timer CompareValue register

A.9 AArch64 Other system control registers summary

The summary table provides an overview of all Other system control registers in the core. For more information about a register, click the register name in the table.

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

Table A-399: Other system control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SCTLR_EL1	3	0	C1	C0	0	–	64-bit	System Control Register (EL1)
CPACR_EL1	3	0	C1	C0	2	–	64-bit	Architectural Feature Access Control Register
ZCR_EL1	3	0	C1	C2	0	–	64-bit	SVE Control Register (EL1)
SCTLR_EL2	3	4	C1	C0	0	–	64-bit	System Control Register (EL2)
HCR_EL2	3	4	C1	C1	0	–	64-bit	Hypervisor Configuration Register
CPTR_EL2	3	4	C1	C1	2	–	64-bit	Architectural Feature Trap Register (EL2)
HSTR_EL2	3	4	C1	C1	3	–	64-bit	Hypervisor System Trap Register
ZCR_EL2	3	4	C1	C2	0	–	64-bit	SVE Control Register (EL2)
SCTLR_EL3	3	6	C1	C0	0	–	64-bit	System Control Register (EL3)
ZCR_EL3	3	6	C1	C2	0	–	64-bit	SVE Control Register (EL3)

A.10 AArch64 Activity Monitors registers summary

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

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

Table A-400: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	–	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	–	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	–	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	–	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	–	64-bit	Activity Monitors Count Enable Clear Register 0
AMCNTENSET0_EL0	3	3	C13	C2	5	–	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	–	64-bit	Activity Monitors Count Enable Clear Register 1
AMCNTENSET1_EL0	3	3	C13	C3	1	–	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	–	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	–	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	–	64-bit	Activity Monitors Event Counter Registers 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMEVCNTR03_EL0	3	3	C13	C4	3	–	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	–	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	–	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	–	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	–	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	–	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	–	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	–	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	–	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	–	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	–	64-bit	Activity Monitors Event Type Registers 1

A.10.1 AMCFGR_EL0, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR_EL0 is applicable to both the architected and the auxiliary counter groups.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0001 xxx1 0000 0000 0011 1111 0000 0110


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-161: AArch64_amcfgr_el0 bit assignments

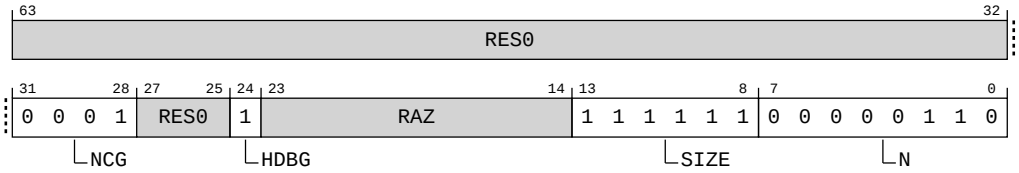


Table A-401: AMCFGR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. **0b0001** Two counter groups are implemented	0b0001
[27:25]	**RES0**	Reserved	**RES0**
[24]	HDBG	Halt-on-debug supported. This feature must be supported, and so this bit is 0b1. **0b1** AArch64-AMCR_EL0.HDBG is read/write.	0b1
[23:14]	**RAZ**	Reserved	**RAZ**
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the activity monitors Extension is [AMCFGR_EL0.SIZE + 1]. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. **0b111111** 64 bits.	0b111111
[7:0]	N	Defines the number of activity monitor event counters. The total number of counters implemented in all groups by the Activity Monitors Extension is [AMCFGR_EL0.N + 1]. **0b00000110** Seven activity monitor event counters	0x06

Access

MRS <Xt>, AMCFGR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b001

Accessibility

MRS <Xt>, AMCFGR_EL0

```
if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCFGR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCFGR_EL0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCFGR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMCFGR_EL0;
```

A.10.2 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0011 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-162: AArch64_amcgcr_el0 bit assignments

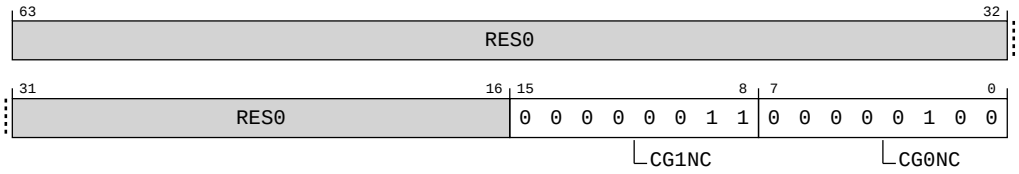


Table A-403: AMCGCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. In an implementation that includes FEAT_AMUv1, the permitted range of values is 0x0 to 0x10. **0b00000011** Three counters in the auxiliary counter group	0x03
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. **0b00000100** Four Counters in the architected counter group	0x04

Access

MRS <Xt>, AMCGCR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0010	0b010

Accessibility

MRS <Xt>, AMCGCR_EL0

```
if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCGCR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCGCR_EL0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMCGCR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMCGCR_EL0;
```

A.10.3 AMEVCNTR00_EL0, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-163: AArch64_amevcntr00_el0 bit assignments

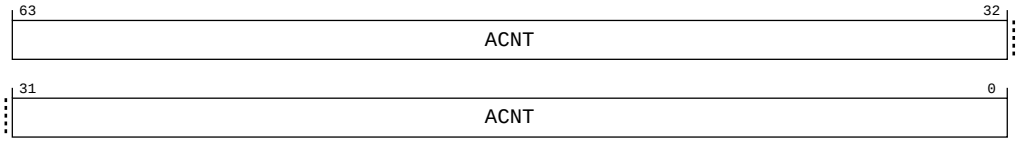


Table A-405: AMEVCNTR00_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 0.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR00_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b000

MSR AMEVCNTR00_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b000

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are UNDEFINED.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR00_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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[0];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR0_EL0[0];
```

MSR AMEVCNTR00_EL0, <Xt>

```
if IsHighestEL(PSTATE.EL) then
    AMEVCNTR0_EL0[0] = X[t, 64];
else
    UNDEFINED;
```

A.10.4 AMEVCNTR01_EL0, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-164: AArch64_amevcntr01_el0 bit assignments

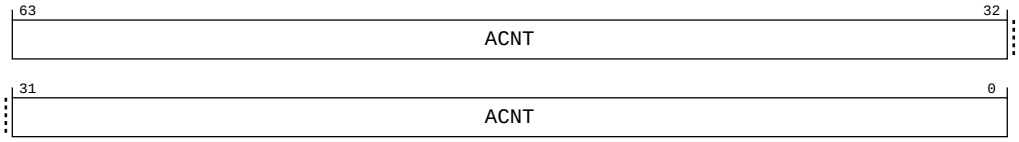


Table A-408: AMEVCNTR01_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 1.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR01_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b001

MSR AMEVCNTR01_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b001

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are UNDEFINED.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR01_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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[1];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR0_EL0[1];
```

MSR AMEVCNTR01_EL0, <Xt>

```
if IsHighestEL(PSTATE.EL) then
    AMEVCNTR0_EL0[1] = X[t, 64];
else
    UNDEFINED;
```

A.10.5 AMEVCNTR02_EL0, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-165: AArch64_amevcntr02_el0 bit assignments

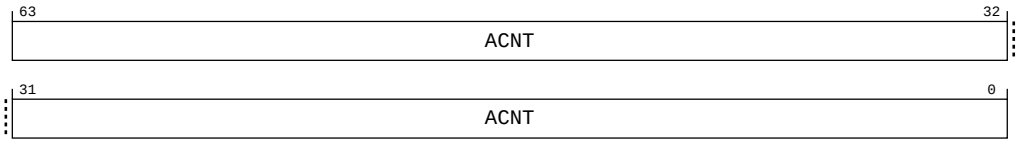


Table A-411: AMEVCNTR02_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 2.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are **UNDEFINED**.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR02_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b010

MSR AMEVCNTR02_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b010

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR02_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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[2];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR0_EL0[2];
```

MSR AMEVCNTR02_EL0, <Xt>

```
if IsHighestEL(PSTATE.EL) then
    AMEVCNTR0_EL0[2] = X[t, 64];
else
    UNDEFINED;
```

A.10.6 AMEVCNTR03_EL0, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counter 3.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-166: AArch64_amevcntr03_el0 bit assignments

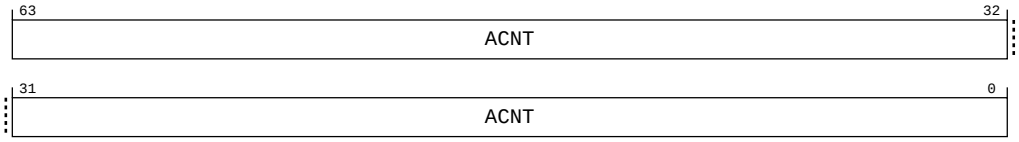


Table A-414: AMEVCNTR03_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of architected activity monitor event counter 3.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR03_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b011

MSR AMEVCNTR03_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0100	0b011

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVCNTR03_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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[3];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[3];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR0_EL0[3];
```

MSR AMEVCNTR03_EL0, <Xt>

```
if IsHighestEL(PSTATE.EL) then
    AMEVCNTR0_EL0[3] = X[t, 64];
else
    UNDEFINED;
```

A.10.7 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 registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-167: AArch64_amevtyper00_el0 bit assignments

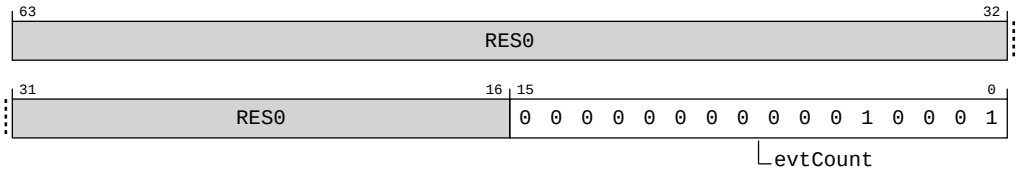


Table A-417: AMEVTYPER00_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR00_EL0. The value of this field is architecturally mandated for each architected counter. **0b0000000000010001** Processor frequency cycles	0x0011

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER00_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b000

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER00_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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[0];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[0];
```

A.10.8 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 registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-168: AArch64_amevtyper01_el0 bit assignments

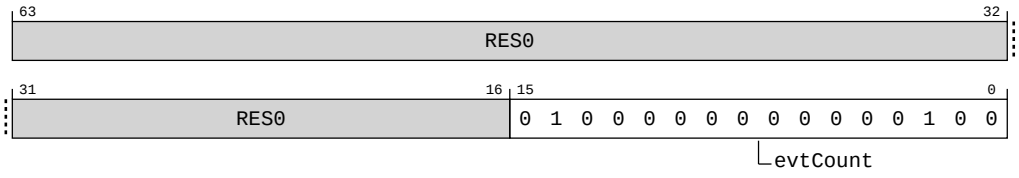


Table A-419: AMEVTYPER01_EL0 bit descriptions

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

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR01_EL0. The value of this field is architecturally mandated for each architected counter. **0b0100000000000100** Constant frequency cycles	0x4004

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER01_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b001

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER01_EL0

```
if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[1];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[1];
```

A.10.9 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 registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 1000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-169: AArch64_amevtyper02_el0 bit assignments

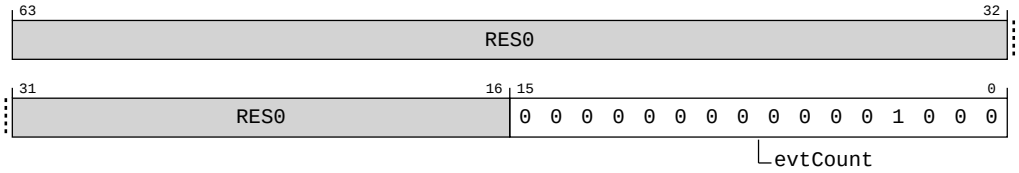


Table A-421: AMEVTYPER02_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR02_EL0. The value of this field is architecturally mandated for each architected counter. **0b0000000000001000** Instructions retired	0x0008

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER02_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b010

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER02_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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[2];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[2];
```

A.10.10 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 registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0100 0000 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-170: AArch64_amevtyper03_el0 bit assignments

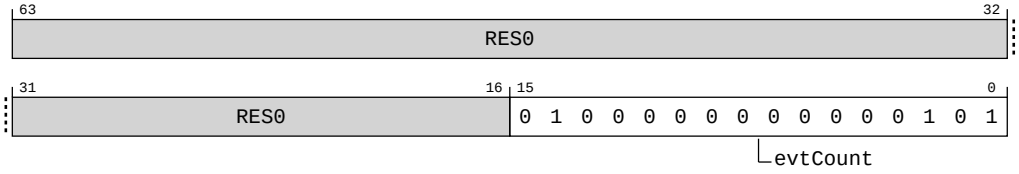


Table A-423: AMEVTYPER03_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR03_EL0. The value of this field is architecturally mandated for each architected counter. **0b0100000000000101** Memory stall cycles	0x4005

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER03_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b011

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER03_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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[3];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER0_EL0[3];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER0_EL0[3];
```

A.10.11 AMEVCNTR10_EL0, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-171: AArch64_amevcntr10_el0 bit assignments

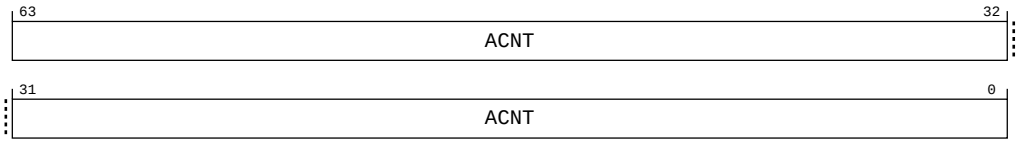


Table A-425: AMEVCNTR10_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of Auxiliary activity monitor event counter 0.	0x0000000000000000

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVCNTR10_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1100	0b000

MSR AMEVCNTR10_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1100	0b000

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVCNTR10_EL0

```
if 0 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[0];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR1_EL0[0];
```

MSR AMEVCNTR10_EL0, <Xt>

```
if 0 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif IsHighestEL(PSTATE.EL) then
    AMEVCNTR1_EL0[0] = X[t, 64];
else
    UNDEFINED;
```

A.10.12 AMEVCNTR11_EL0, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-172: AArch64_amevcntr11_el0 bit assignments

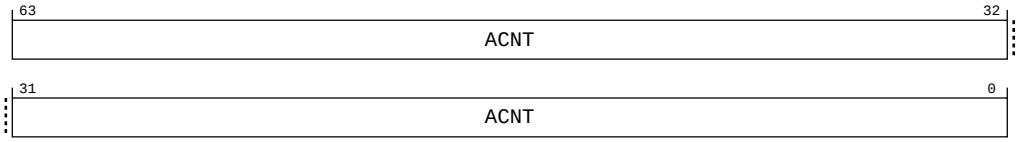


Table A-428: AMEVCNTR11_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of Auxiliary activity monitor event counter 1.	0x0000000000000000

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVCNTR11_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1100	0b001

MSR AMEVCNTR11_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1100	0b001

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVCNTR11_EL0

```
if 1 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[1];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR1_EL0[1];
```

MSR AMEVCNTR11_EL0, <Xt>

```
if 1 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif IsHighestEL(PSTATE.EL) then
    AMEVCNTR1_EL0[1] = X[t, 64];
else
    UNDEFINED;
```

A.10.13 AMEVCNTR12_EL0, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counter 2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors registers

Access type

See bit descriptions

Reset value

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

Bit descriptions

Figure A-173: AArch64_amevcntr12_el0 bit assignments

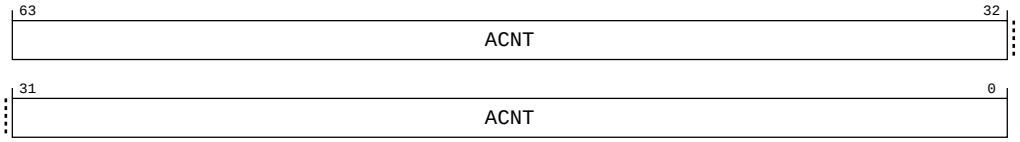


Table A-431: AMEVCNTR12_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Value of Auxiliary activity monitor event counter 2.	0x0000000000000000

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n>_EL0 are **UNDEFINED**.


Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVCNTR12_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1100	0b010

MSR AMEVCNTR12_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1100	0b010

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVCNTR1<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVCNTR12_EL0

```
if 2 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[2];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVCNTR1_EL0[2];
```

MSR AMEVCNTR12_EL0, <Xt>

```
if 2 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif IsHighestEL(PSTATE.EL) then
    AMEVCNTR1_EL0[2] = X[t, 64];
else
    UNDEFINED;
```

A.10.14 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 registers

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-174: AArch64_amevtyper10_el0 bit assignments

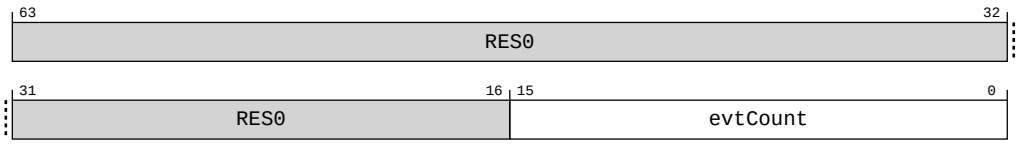


Table A-434: AMEVTYPER10_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR1<n>_EL0 **0b0000001100000000** Gear 0 (MPMM bank 0) period threshold exceeded	`16{x}`

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER10_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b000

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are UNDEFINED.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER10_EL0

```
if 0 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[0];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[0];
```

A.10.15 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 registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-175: AArch64_amevtyper11_el0 bit assignments

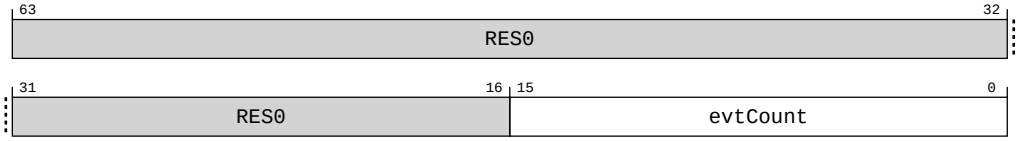


Table A-436: AMEVTYPER11_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR1<n>_EL0 **0b0000001100000001** Gear 1 (MPMM bank 1) period threshold exceeded	16{x}

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER11_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b001

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are UNDEFINED.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER11_EL0

```
if 1 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[1];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[1];
```

A.10.16 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 registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-176: AArch64_amevtyper12_el0 bit assignments

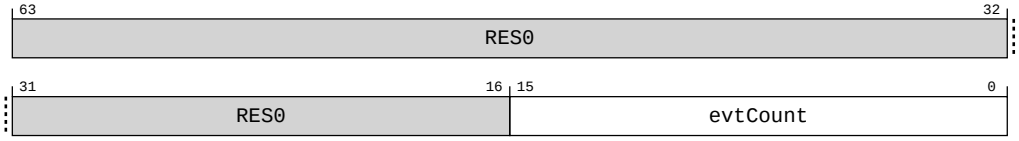


Table A-438: AMEVTYPER12_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR1<n>_EL0 **0b0000001100000010** Gear 2 (MPMM bank 2) period threshold exceeded	16{x}

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are **UNDEFINED**.

Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER12_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b1110	0b010

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads and writes of AMEVTYPER1<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG1NC identifies the number of auxiliary activity monitor event counters.

MRS <Xt>, AMEVTYPER12_EL0

```
if 2 >= NUM_AMU_CG1_MONITORS then
    UNDEFINED;
elsif 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
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[2];
elsif PSTATE.EL == EL3 then
    X[t, 64] = AMEVTYPER1_EL0[2];
```

A.11 AArch64 Trace unit registers summary

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

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

Table A-440: Trace unit registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCTRACEIDR	2	1	C0	C0	1	—	64-bit	Trace ID Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCVICTLR	2	1	C0	C0	2	–	64-bit	ViewInst Main Control Register
TRCSEQEVR0	2	1	C0	C0	4	–	64-bit	Sequencer State Transition Control Register <n>
TRCIDR8	2	1	C0	C0	6	–	64-bit	ID Register 8
TRCIMSPEC0	2	1	C0	C0	7	–	64-bit	IMP DEF Register 0
TRCPRGCTLR	2	1	C0	C1	0	–	64-bit	Programming Control Register
TRCVIIECTLR	2	1	C0	C1	2	–	64-bit	ViewInst Include/Exclude Control Register
TRCSEQEVR1	2	1	C0	C1	4	–	64-bit	Sequencer State Transition Control Register <n>
TRCIDR9	2	1	C0	C1	6	–	64-bit	ID Register 9
TRCVISSCTLR	2	1	C0	C2	2	–	64-bit	ViewInst Start/Stop Control Register
TRCSEQEVR2	2	1	C0	C2	4	–	64-bit	Sequencer State Transition Control Register <n>
TRCIDR10	2	1	C0	C2	6	–	64-bit	ID Register 10
TRCSTATR	2	1	C0	C3	0	–	64-bit	Trace Status Register
TRCIDR11	2	1	C0	C3	6	–	64-bit	ID Register 11
TRCCONFIGR	2	1	C0	C4	0	–	64-bit	Trace Configuration Register
TRCCNTCTLR0	2	1	C0	C4	5	–	64-bit	Counter Control Register <n>
TRCIDR12	2	1	C0	C4	6	–	64-bit	ID Register 12
TRCCNTCTLR1	2	1	C0	C5	5	–	64-bit	Counter Control Register <n>
TRCIDR13	2	1	C0	C5	6	–	64-bit	ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	–	64-bit	Auxiliary Control Register
TRCSEQRSTEVR	2	1	C0	C6	4	–	64-bit	Sequencer Reset Control Register
TRCSEQSTR	2	1	C0	C7	4	–	64-bit	Sequencer State Register
TRCEVENTCTL0R	2	1	C0	C8	0	–	64-bit	Event Control 0 Register
TRCEXTINSELR0	2	1	C0	C8	4	–	64-bit	External Input Select Register <n>
TRCCNTVR0	2	1	C0	C8	5	–	64-bit	Counter Value Register <n>
TRCIDR0	2	1	C0	C8	7	–	64-bit	ID Register 0
TRCEVENTCTL1R	2	1	C0	C9	0	–	64-bit	Event Control 1 Register
TRCEXTINSELR1	2	1	C0	C9	4	–	64-bit	External Input Select Register <n>
TRCCNTVR1	2	1	C0	C9	5	–	64-bit	Counter Value Register <n>
TRCIDR1	2	1	C0	C9	7	–	64-bit	ID Register 1
TRCRSR	2	1	C0	C10	0	–	64-bit	Resources Status Register
TRCEXTINSELR2	2	1	C0	C10	4	–	64-bit	External Input Select Register <n>
TRCIDR2	2	1	C0	C10	7	–	64-bit	ID Register 2
TRCEXTINSELR3	2	1	C0	C11	4	–	64-bit	External Input Select Register <n>
TRCIDR3	2	1	C0	C11	7	–	64-bit	ID Register 3
TRCTSCTLR	2	1	C0	C12	0	–	64-bit	Timestamp Control Register
TRCIDR4	2	1	C0	C12	7	–	64-bit	ID Register 4
TRCSYNCPR	2	1	C0	C13	0	–	64-bit	Synchronization Period Register
TRCIDR5	2	1	C0	C13	7	–	64-bit	ID Register 5
TRCCCCTLR	2	1	C0	C14	0	–	64-bit	Cycle Count Control Register
TRCIDR6	2	1	C0	C14	7	–	64-bit	ID Register 6

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCBBCTLR	2	1	C0	C15	0	–	64-bit	Branch Broadcast Control Register
TRCIDR7	2	1	C0	C15	7	–	64-bit	ID Register 7
TRCSSCCR0	2	1	C1	C0	2	–	64-bit	Single-shot Comparator Control Register <n>
TRCOSLSR	2	1	C1	C1	4	–	64-bit	Trace OS Lock Status Register
TRCRSCTLR2	2	1	C1	C2	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR3	2	1	C1	C3	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR4	2	1	C1	C4	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR5	2	1	C1	C5	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR6	2	1	C1	C6	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR7	2	1	C1	C7	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR8	2	1	C1	C8	0	–	64-bit	Resource Selection Control Register <n>
TRCSSCSR0	2	1	C1	C8	2	–	64-bit	Single-shot Comparator Control Status Register <n>
TRCRSCTLR9	2	1	C1	C9	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR10	2	1	C1	C10	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR11	2	1	C1	C11	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR12	2	1	C1	C12	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR13	2	1	C1	C13	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR14	2	1	C1	C14	0	–	64-bit	Resource Selection Control Register <n>
TRCRSCTLR15	2	1	C1	C15	0	–	64-bit	Resource Selection Control Register <n>
TRCACVR0	2	1	C2	C0	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR0	2	1	C2	C0	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR1	2	1	C2	C2	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR1	2	1	C2	C2	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR2	2	1	C2	C4	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR2	2	1	C2	C4	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR3	2	1	C2	C6	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR3	2	1	C2	C6	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR4	2	1	C2	C8	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR4	2	1	C2	C8	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR5	2	1	C2	C10	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR5	2	1	C2	C10	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR6	2	1	C2	C12	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR6	2	1	C2	C12	2	–	64-bit	Address Comparator Access Type Register <n>
TRCACVR7	2	1	C2	C14	0	–	64-bit	Address Comparator Value Register <n>
TRCACATR7	2	1	C2	C14	2	–	64-bit	Address Comparator Access Type Register <n>
TRCCIDCVR0	2	1	C3	C0	0	–	64-bit	Context Identifier Comparator Value Registers <n>
TRCVMIDCVR0	2	1	C3	C0	1	–	64-bit	Virtual Context Identifier Comparator Value Register <n>
TRCCIDCCTLR0	2	1	C3	C0	2	–	64-bit	Context Identifier Comparator Control Register 0
TRCVMIDCCTLR0	2	1	C3	C2	2	–	64-bit	Virtual Context Identifier Comparator Control Register 0
TRCDEVID	2	1	C7	C2	7	–	64-bit	Device Configuration Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCCLAIMSET	2	1	C7	C8	6	–	64-bit	Claim Tag Set Register
TRCCLAIMCLR	2	1	C7	C9	6	–	64-bit	Claim Tag Clear Register
TRCAUTHSTATUS	2	1	C7	C14	6	–	64-bit	Authentication Status Register
TRCDEVARCH	2	1	C7	C15	6	–	64-bit	Device Architecture Register

A.11.1 TRCSEQEVR0, Sequencer State Transition Control Register <n>

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-177: AArch64_trcseqevr0 bit assignments

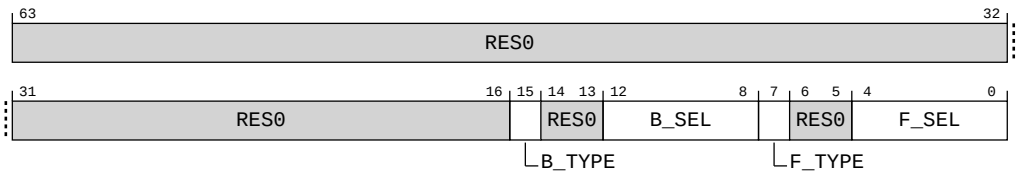


Table A-441: TRCSEQEVR0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	B_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == `0x14` then when event `0x14` occurs, the Sequencer moves from state 3 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	B_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Backward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}
[7]	F_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == `0x12` then when event `0x12` occurs, the Sequencer moves from state 1 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4:0]	F_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Forward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCSEQEVR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b100

MSR TRCSEQEVR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b100

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCSEQEVR0

```
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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[0];
```

MSR TRCSEQEVR0, <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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[0] = X[t, 64];
```

A.11.2 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

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-178: AArch64_trcidr8 bit assignments

Table A-444: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time. **0b00000000000000000000000000000000** No speculation in the trace element stream	0x00000000

Access

MRS <Xt>, TRCIDR8

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b110

Accessibility

MRS <Xt>, TRCIDR8

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR8;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR8;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR8;
```

A.11.3 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 unit registers

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-179: AArch64_trcimspec0 bit assignments

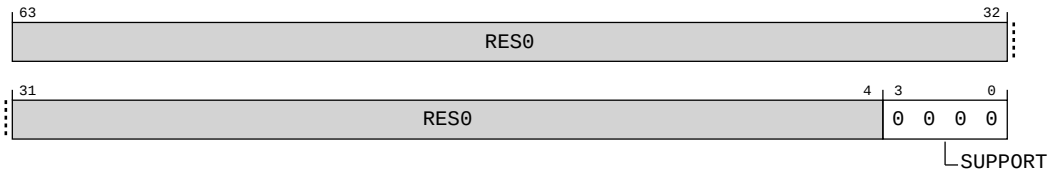


Table A-446: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[63:4]	**RES0**	Reserved	**RES0**
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No **IMPLEMENTATION DEFINED** features are supported.	0b0000

Access

MRS <Xt>, TRCIMSPEC0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

MSR TRCIMSPEC0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, TRCIMSPEC0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        X[t, 64] = TRCIMSPEC0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIMSPEC0;
```

MSR TRCIMSPEC0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 64];
```

A.11.4 TRCSEQEVR1, Sequencer State Transition Control Register <n>

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-180: AArch64_trcseqevr1 bit assignments

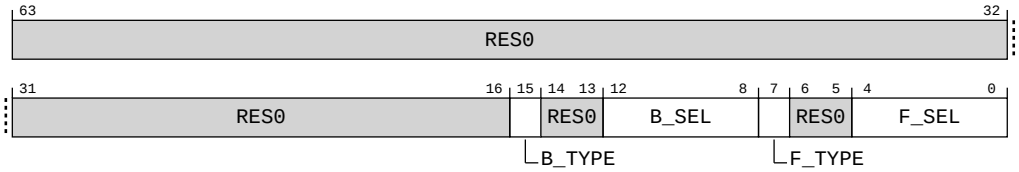


Table A-449: TRCSEQEVR1 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	B_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == `0x14` then when event `0x14` occurs, the Sequencer moves from state 3 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[12:8]	B_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Backward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}
[7]	F_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == 0x12 then when event 0x12 occurs, the Sequencer moves from state 1 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	F_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Forward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCSEQEVR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b100

MSR TRCSEQEVR1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b100

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCSEQEVR1

```
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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[1];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[1];
```

MSR TRCSEQEVR1, <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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[1] = X[t, 64];
```

A.11.5 TRCSEQEVR2, Sequencer State Transition Control Register <n>

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-181: AArch64_trcseqevr2 bit assignments

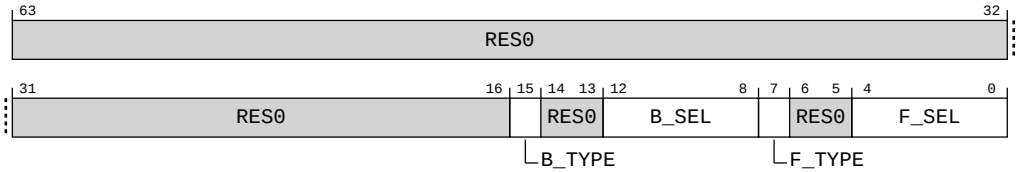


Table A-452: TRCSEQEVR2 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	B_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == `0x14` then when event `0x14` occurs, the Sequencer moves from state 3 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is **RES0**.	`x`
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	B_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Backward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	`5{x}`

Bits	Name	Description	Reset
[7]	F_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == 0x12 then when event 0x12 occurs, the Sequencer moves from state 1 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	F_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Forward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCSEQEVR2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b100

MSR TRCSEQEVR2, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b100

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCSEQEVR2

```
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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[2];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSEQEVR[2];
```

MSR TRCSEQEVR2, <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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSEQEVR[2] = X[t, 64];
```

A.11.6 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 unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-182: AArch64_trcidr10 bit assignments

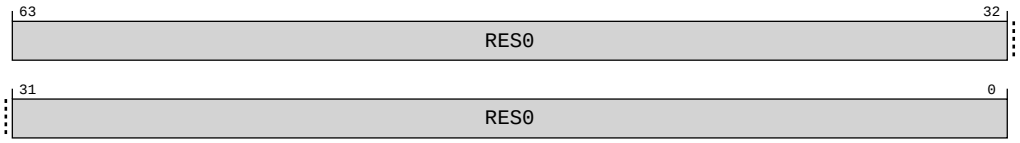


Table A-455: TRCIDR10 bit descriptions

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

Access

MRS <Xt>, TRCIDR10

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b110

Accessibility

MRS <Xt>, TRCIDR10

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR10;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR10;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR10;
```

A.11.7 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 unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-183: AArch64_trcidr11 bit assignments

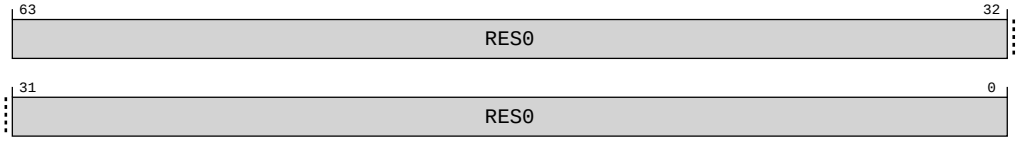


Table A-457: TRCIDR11 bit descriptions

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

Access

MRS <Xt>, TRCIDR11

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b110

Accessibility

MRS <Xt>, TRCIDR11

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR11;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR11;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR11;
```

A.11.8 TRCCNTCTLR0, Counter Control Register <n>

Controls the operation of Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-184: AArch64_trccntctlr0 bit assignments

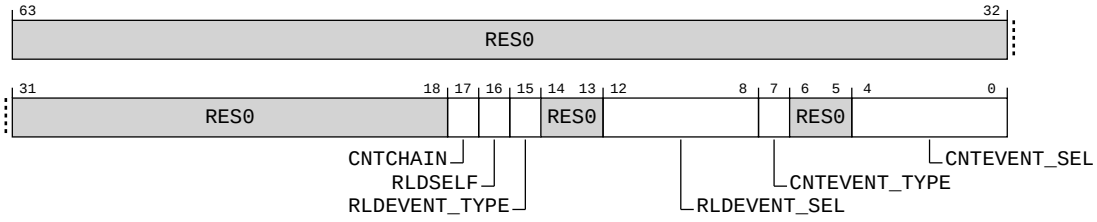


Table A-459: TRCCNTCTLR0 bit descriptions

Bits	Name	Description	Reset
[63:18]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[17]	CNTCHAIN	For TRCCNTCTLR3 and TRCCNTCTLR1, this field controls whether the Counter decrements when a reload event occurs for Counter <n-1>. **0b0** The Counter does not decrement when a reload event for Counter <n-1> occurs. **0b1** Counter <n> decrements when a reload event for Counter <n-1> occurs. This concatenates Counter <n> and Counter <n-1>, to provide a larger count value. CNTCHAIN is not implemented for TRCCNTCTLR0 and TRCCNTCTLR2.	x
[16]	RLDSELF	Controls whether a reload event occurs for the Counter, when the Counter reaches zero. **0b0** Normal mode. The Counter is in Normal mode. **0b1** Self-reload mode. The Counter is in Self-reload mode.	x
[15]	RLDEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes a reload event for Counter <n>. **0b0** A single Resource Selector. TRCCNTCTLR<n>.RLDEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.RLDEVENT.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	RLDEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.TYPE controls whether TRCCNTCTLR<n>.RLDEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes a reload event for Counter <n>. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Bits	Name	Description	Reset
[7]	CNTEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes Counter <n> to decrement. **0b0** A single Resource Selector. TRCCNTCTLR<n>.CNTEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.CNTEVENT.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	CNTEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.TYPE controls whether TRCCNTCTLR<n>.CNTEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes Counter <n> to decrement. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCCNTCTLR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b101

MSR TRCCNTCTLR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b101

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCCNTCTLR0

```
if 0 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[0];
```

MSR TRCCNTCTLR0, <Xt>

```
if 0 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTCTLR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTCTLR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        TRCCNTCTLR[0] = X[t, 64];
```

A.11.9 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 unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-185: AArch64_trcidr12 bit assignments

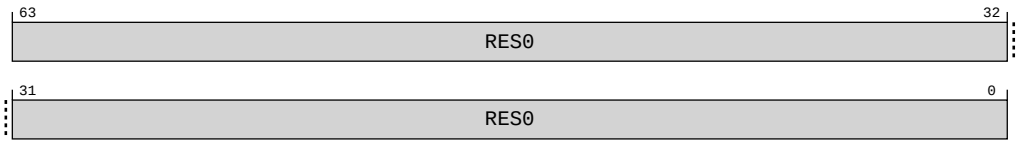


Table A-462: TRCIDR12 bit descriptions

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

Access

MRS <Xt>, TRCIDR12

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b110

Accessibility

MRS <Xt>, TRCIDR12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR12;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR12;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR12;
```

A.11.10 TRCCNTCTLR1, Counter Control Register <n>

Controls the operation of Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-186: AArch64_trccntctlr1 bit assignments

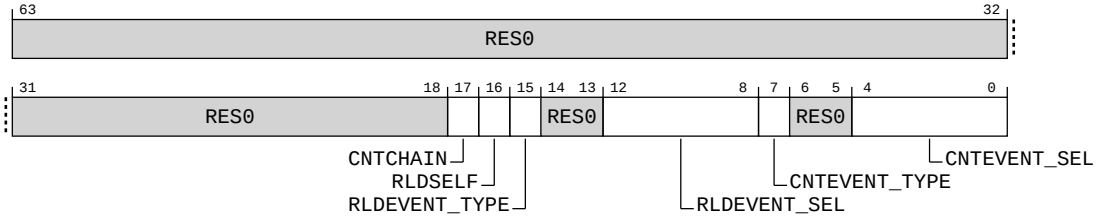


Table A-464: TRCCNTCTLR1 bit descriptions

Bits	Name	Description	Reset
[63:18]	**RES0**	Reserved	**RES0**
[17]	CNTCHAIN	For TRCCNTCTLR3 and TRCCNTCTLR1, this field controls whether the Counter decrements when a reload event occurs for Counter <n-1>. **0b0** The Counter does not decrement when a reload event for Counter <n-1> occurs. **0b1** Counter <n> decrements when a reload event for Counter <n-1> occurs. This concatenates Counter <n> and Counter <n-1>, to provide a larger count value. CNTCHAIN is not implemented for TRCCNTCTLR0 and TRCCNTCTLR2.	x
[16]	RLDSELF	Controls whether a reload event occurs for the Counter, when the Counter reaches zero. **0b0** Normal mode. The Counter is in Normal mode. **0b1** Self-reload mode. The Counter is in Self-reload mode.	x

Bits	Name	Description	Reset
[15]	RLDEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes a reload event for Counter <n>. **0b0** A single Resource Selector. TRCCNTCTLR<n>.RLDEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.RLDEVENT.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	RLDEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.TYPE controls whether TRCCNTCTLR<n>.RLDEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes a reload event for Counter <n>. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}
[7]	CNTEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes Counter <n> to decrement. **0b0** A single Resource Selector. TRCCNTCTLR<n>.CNTEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.CNTEVENT.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4:0]	CNTEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.TYPE controls whether TRCCNTCTLR<n>.CNTEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes Counter <n> to decrement. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCCNTCTLR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0101	0b101

MSR TRCCNTCTLR1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0101	0b101

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCCNTCTLR1

```
if 1 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[1];
elsif PSTATE.EL == EL2 then
```

```
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[1];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[1];
```

MSR TRCCNTCTLR1, <Xt>

```
if 1 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTCTLR[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTCTLR[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTCTLR[1] = X[t, 64];
```

A.11.11 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 unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-187: AArch64_trcidr13 bit assignments

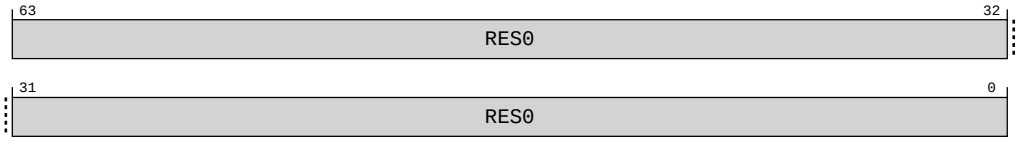


Table A-467: TRCIDR13 bit descriptions

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

Access

MRS <Xt>, TRCIDR13

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0101	0b110

Accessibility

MRS <Xt>, TRCIDR13

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR13;
```

```
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR13;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR13;
```

A.11.12 TRCEXTINSELR0, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-188: AArch64_trcextinselr0 bit assignments

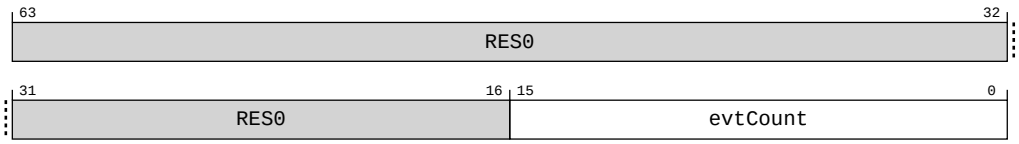


Table A-469: TRCEXTINSELR0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == `0b0000` and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCEXTINSELR0

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0000`	`0b1000`	`0b100`

MSR TRCEXTINSELR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b100

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCEXTINSELR0

```
if 0 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[0];
```

MSR TRCEXTINSELR0, <Xt>

```
if 0 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCEXTINSELR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[0] = X[t, 64];
```

A.11.13 TRCCNTVR0, Counter Value Register <n>

This sets or returns the value of Counter 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-189: AArch64_trccntvr0 bit assignments

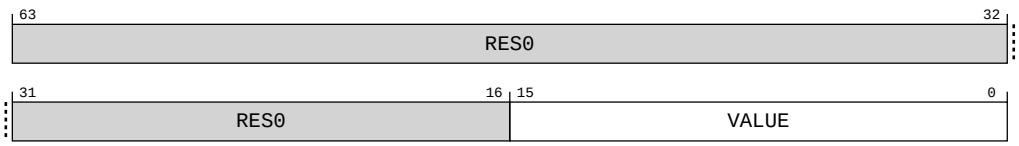


Table A-472: TRCCNTVR0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	VALUE	Contains the count value of Counter.	`16{x}`

Access

Must be programmed if TRCRSCTLR<a>.GROUP == `0b0010` and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

MRS <Xt>, TRCCNTVR0

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0000`	`0b1000`	`0b101`

MSR TRCCNTVR0, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0000`	`0b1000`	`0b101`

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.
MRS <Xt>, TRCCNTVR0

```
if 0 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTVR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTVR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTVR[0];
```

MSR TRCCNTVR0, <Xt>

```
if 0 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTVR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTVR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTVR[0] = X[t, 64];
```

A.11.14 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 unit registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x010 1000 xxxx xxx0 00xx 111x 1010 000x

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-190: AArch64_trcidr0 bit assignments

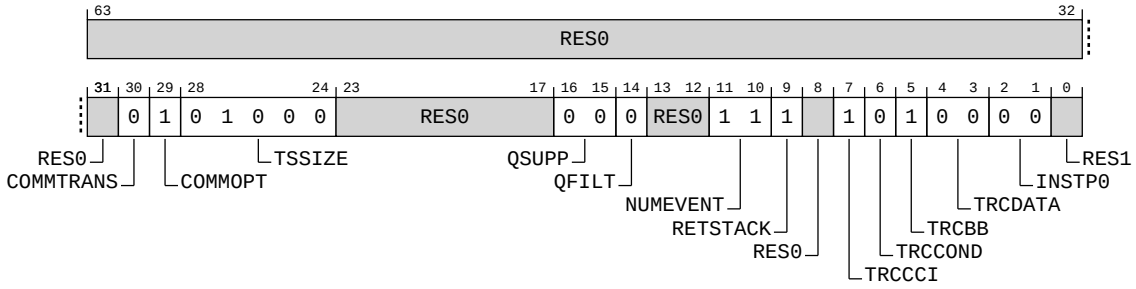


Table A-475: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	0b0
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1.	0b1
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	0b01000
[23:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	0b00

Bits	Name	Description	Reset
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	0b0
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	0b11
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	0b1
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	0b1
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	0b0
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	0b1
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Tracing of data addresses and data values is not implemented.	0b00
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Load and store instructions are not P0 instructions.	0b00
[0]	**RES1**	Reserved	**RES1**

Access

MRS <Xt>, TRCIDR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b111

Accessibility

MRS <Xt>, TRCIDR0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR0;
```

A.11.15 TRCEXTINSELR1, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-191: AArch64_trcextinselr1 bit assignments

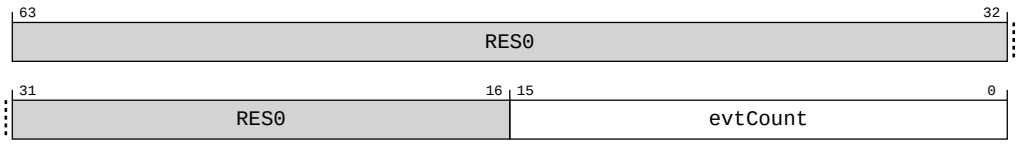


Table A-477: TRCEXTINSELR1 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == `0b0000` and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCEXTINSELR1

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0000`	`0b1001`	`0b100`

MSR TRCEXTINSELR1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b100

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCEXTINSELR1

```
if 1 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[1];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[1];
```

MSR TRCEXTINSELR1, <Xt>

```
if 1 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCEXTINSELR[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[1] = X[t, 64];
```

A.11.16 TRCCNTVR1, Counter Value Register <n>

This sets or returns the value of Counter 1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-192: AArch64_trccntvr1 bit assignments

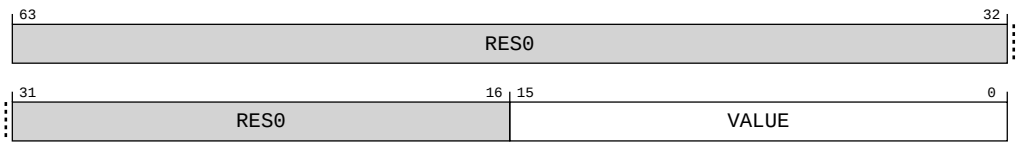


Table A-480: TRCCNTVR1 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	VALUE	Contains the count value of Counter.	`16{x}`

Access

Must be programmed if TRCRSCTLR<a>.GROUP == `0b0010` and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

MRS <Xt>, TRCCNTVR1

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0000`	`0b1001`	`0b101`

MSR TRCCNTVR1, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0000`	`0b1001`	`0b101`

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.
MRS <Xt>, TRCCNTVR1

```
if 1 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTVR[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTVR[1];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTVR[1];
```

MSR TRCCNTVR1, <Xt>

```
if 1 >= NUM_TRACE_COUNTERS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTVR[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTVR[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTVR[1] = X[t, 64];
```

A.11.17 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 unit registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0100 0001 xxxx xxxx xxxx 1111 1111 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-193: AArch64_trcidr1 bit assignments

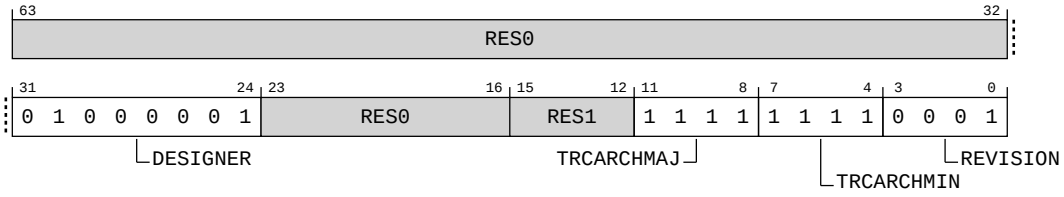


Table A-483: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. **0b01000001** Arm Limited	0x41
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	**RES1**	Reserved	**RES1**
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	0b1111
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	0b1111
[3:0]	REVISION	Implementation revision that identifies the revision of the trace and OS Lock registers. **0b0001** Revision 1	0b0001

Access

MRS <Xt>, TRCIDR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b111

Accessibility

MRS <Xt>, TRCIDR1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR1;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR1;
```

A.11.18 TRCEXTINSELR2, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-194: AArch64_trcextinselr2 bit assignments

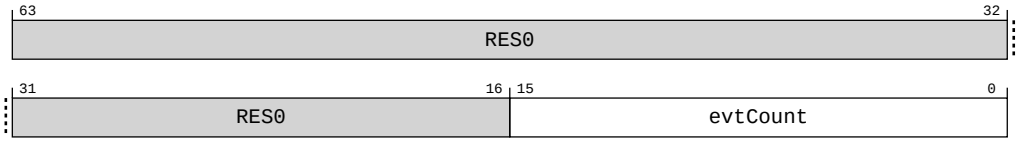


Table A-485: TRCEXTINSELR2 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCEXTINSELR2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b100

MSR TRCEXTINSELR2, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b100

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCEXTINSELR2

```
if 2 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[2];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[2];
```

MSR TRCEXTINSELR2, <Xt>

```
if 2 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[2] = X[t, 64];
```

A.11.19 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 unit registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 1100 000x xxxx xxxx x001 0000 1000 1000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-195: AArch64_trcidr2 bit assignments

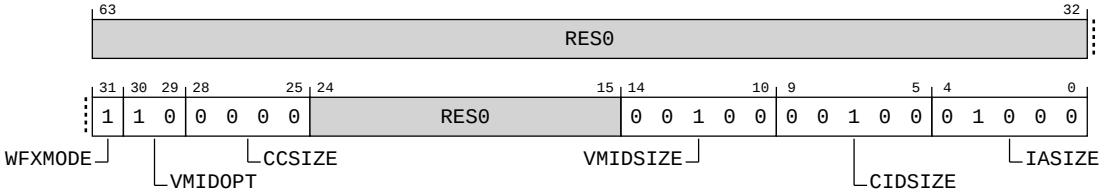


Table A-488: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	WFXMODE	Indicates whether `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions: **0b1** `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions.	0b1
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is **RES1**.	0b10
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	0b0000
[24:15]	**RES0**	Reserved	**RES0**
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	0b00100
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	0b00100
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	0b01000

Access

MRS <Xt>, TRCIDR2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b111

Accessibility

MRS <Xt>, TRCIDR2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR2;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR2;
```

A.11.20 TRCEXTINSELR3, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-196: AArch64_trcextinselr3 bit assignments

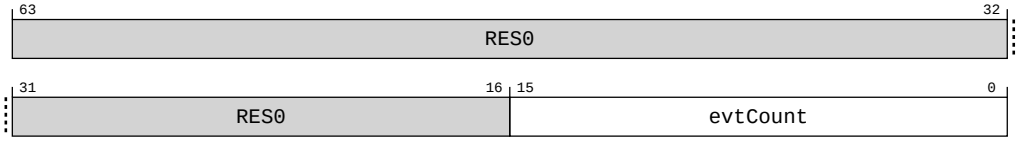


Table A-490: TRCEXTINSELR3 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == `0b0000` and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCEXTINSELR3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b100

MSR TRCEXTINSELR3, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b100

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCEXTINSELR3

```
if 3 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[3];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[3];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCEXTINSELR[3];
```

MSR TRCEXTINSELR3, <Xt>

```
if 3 >= NUM_TRACE_EXTERNAL_INPUT_SELECTOR_RESOURCES then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCEXTINSELR[3] = X[t, 64];
```

A.11.21 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 unit registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0001 x111 1111 xx00 0000 0000 0100


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-197: AArch64_trcidr3 bit assignments

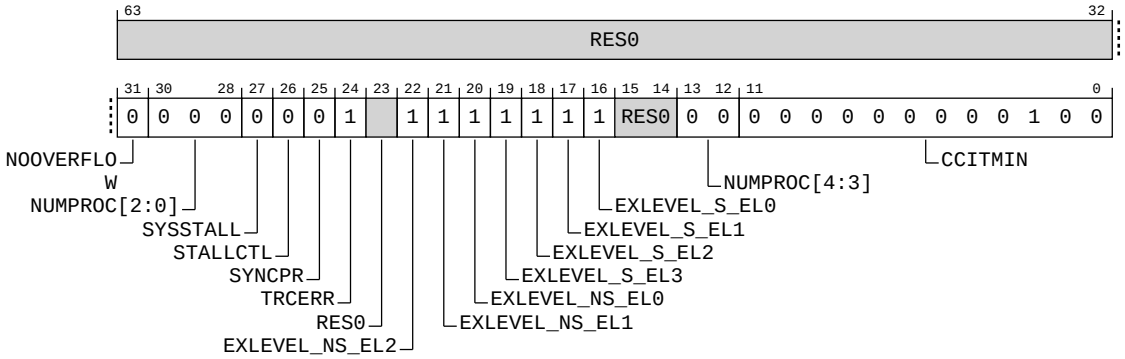


Table A-493: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	0b0
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	0b0
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	0b0
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** AArch64-TRCSYNCPR is read/write so software can change the synchronization period.	0b0
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	0b1
[23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 is implemented. **0b1** Non-secure EL2 is implemented.	0b1
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 is implemented. **0b1** Non-secure EL1 is implemented.	0b1
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 is implemented. **0b1** Non-secure EL0 is implemented.	0b1

Bits	Name	Description	Reset
[19]	EXLEVEL_S_EL3	Indicates if EL3 is implemented. **0b1** EL3 is implemented.	0b1
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 is implemented. **0b1** Secure EL2 is implemented.	0b1
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 is implemented. **0b1** Secure EL1 is implemented.	0b1
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 is implemented. **0b1** Secure EL0 is implemented.	0b1
[15:14]	**RES0**	Reserved	**RES0**
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	0b00000
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in AArch64-TRCCCCTLR.THRESHOLD. If AArch64-TRCIDR0.TRCCCI == 1 then the minimum value of this field is 0x001. If AArch64-TRCIDR0.TRCCCI == 0 then this field is zero. **0b000000000100**	0x004

Access

MRS <Xt>, TRCIDR3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b111

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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR3;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR3;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR3;
```

A.11.22 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 unit registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0001 0001 0001 0111 0000 xxx0 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-198: AArch64_trcidr4 bit assignments

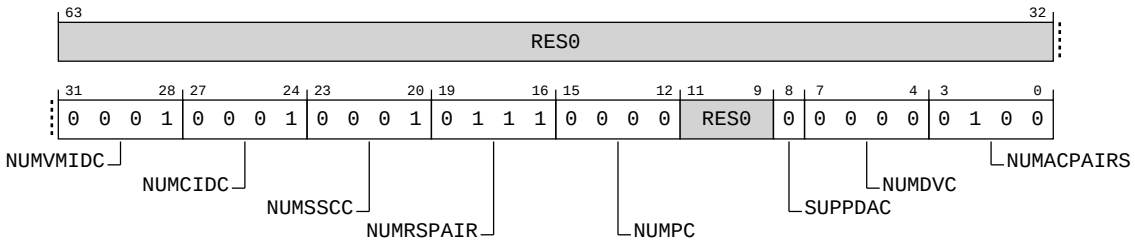


Table A-495: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Virtual Context Identifier Comparator.	0b0001
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	0b0001
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The implementation has one Single-shot Comparator Control.	0b0001
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	0b0111
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	0b0000
[11:9]	**RES0**	Reserved	**RES0**
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** Data address comparisons not implemented.	0b0
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** No data value comparators implemented.	0b0000
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The implementation has four Address Comparator pairs.	0b0100

Access

MRS <Xt>, TRCIDR4

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b111

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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR4;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR4;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR4;
```

A.11.23 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 unit registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x010 100x 0100 0111 xxxx 1001 1111 1111

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-199: AArch64_trcidr5 bit assignments

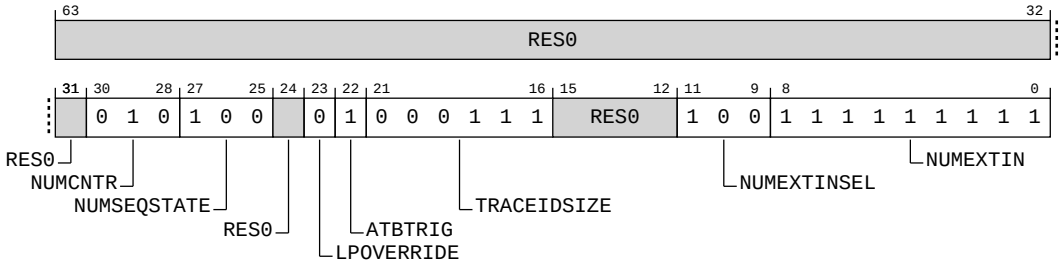


Table A-497: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **0b010** Two Counters implemented.	0b010
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	0b100
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b0** The trace unit does not support Low-power Override Mode.	0b0
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	0b1
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	0b000111
[15:12]	**RES0**	Reserved	**RES0**
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** 4 External Input Selector resources are available.	0b100
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection.	0b111111111

Access

MRS <Xt>, TRCIDR5

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b111

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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR5;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR5;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCIDR5;
```

A.11.24 TRCSSCCR0, Single-shot Comparator Control Register <n>

Controls the corresponding Single-shot Comparator Control resource.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-200: AArch64_trcssccr0 bit assignments

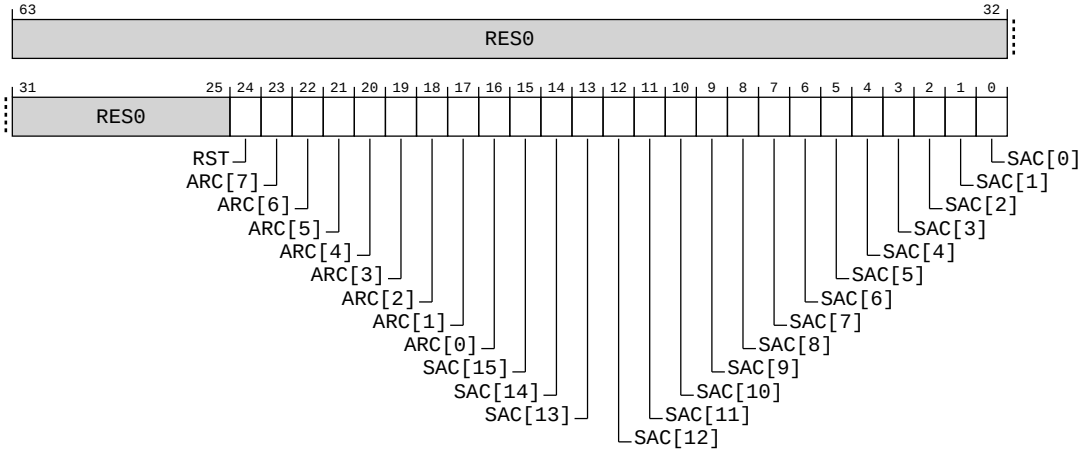


Table A-499: TRCSSCCR0 bit descriptions

Bits	Name	Description	Reset
[63:25]	**RES0**	Reserved	**RES0**
[24]	RST	Selects the Single-shot Comparator Control mode. **0b0** The Single-shot Comparator Control is in single-shot mode. **0b1** The Single-shot Comparator Control is in multi-shot mode.	x
[23:16]	ARC[<m>], bit[m], where m = 7 to 0	Selects one or more Address Range Comparators for Single-shot control. **0b0** The Address Range Comparator <m>, is not selected for Single-shot control. **0b1** The Address Range Comparator <m>, is selected for Single-shot control. This bit is **RES0** if m >= AArch64-TRCIDR4.NUMACPAIRS.	8{x}

Bits	Name	Description	Reset
[15:0]	SAC[<m>], bit[m], where m = 15 to 0	Selects one or more Single Address Comparators for Single-shot control. **0b0** The Single Address Comparator <m>, is not selected for Single-shot control. **0b1** The Single Address Comparator <m>, is selected for Single-shot control. This bit is **RES0** if m >= 2 × AArch64-TRCIDR4.NUMACPAIRS.	`16{x}`

Access

Must be programmed if any TRCRSCTLR<a>.GROUP == `0b0011` and TRCRSCTLR<a>.SINGLE_SHOT[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCSSCCR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0000	0b010

MSR TRCSSCCR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0000	0b010

Accessibility

Must be programmed if any TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCSSCCR0

```
if 0 >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCCR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCCR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCCR[0];
```

MSR TRCSSCCR0, <Xt>

```
if 0 >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSSCCR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSSCCR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSSCCR[0] = X[t, 64];
```

A.11.25 TRCRSCTLR2, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-201: AArch64_trcrsctlr2 bit assignments

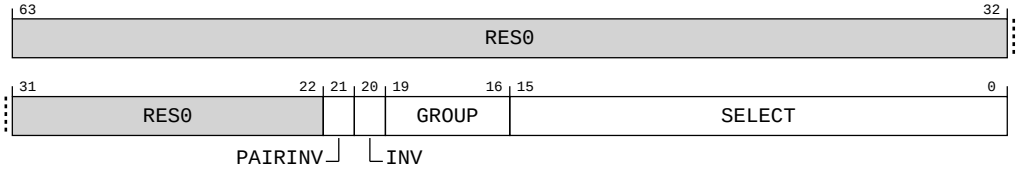


Table A-502: TRCRSCTLR2 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[21]	PAIRINV	Controls whether the combined result from a resource selector pair is inverted. **0b0** Do not invert the combined output of the 2 resource selectors. **0b1** Invert the combined output of the 2 resource selectors. If: • A is the register TRCRSCTLR<n>. • B is the register TRCRSCTLR<n+1>. Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows: • 0b000 -> A and B. • 0b001 -> Reserved. • 0b010 -> not(A) and B. • 0b011 -> not(A) and not(B). • 0b100 -> not(A) or not(B). • 0b101 -> not(A) or B. • 0b110 -> Reserved. • 0b111 -> A or B.	x
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0010	0b000

MSR TRCRSCTLR2, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0010	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR2

```
if 2 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[2];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[2];
```

MSR TRCRSCTLR2, <Xt>

```
if 2 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[2] = X[t, 64];
```

A.11.26 TRCRSCTLR3, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-202: AArch64_trcrsctlr3 bit assignments

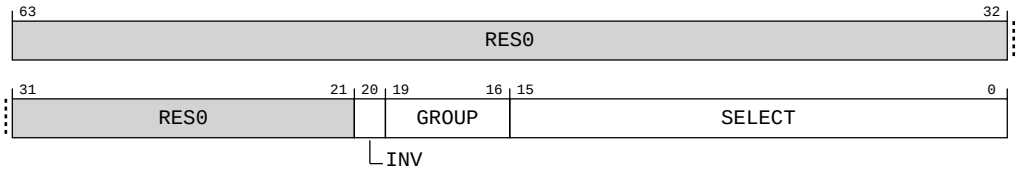


Table A-505: TRCRSCTLR3 bit descriptions

Bits	Name	Description	Reset
[63:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0011	0b000

MSR TRCRSCTLR3, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0011	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR3

```
if 3 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[3];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[3];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[3];
```

MSR TRCRSCTLR3, <Xt>

```
if 3 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[3] = X[t, 64];
```

A.11.27 TRCRSCTLR4, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-203: AArch64_trcrsctlr4 bit assignments

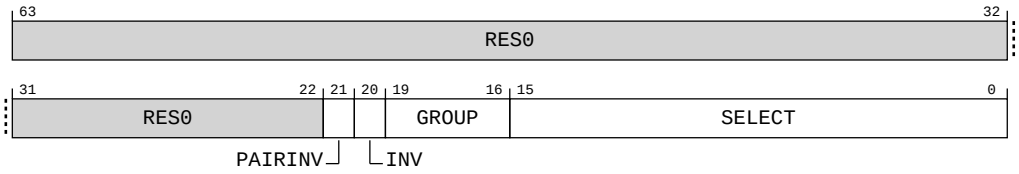


Table A-508: TRCRSCTLR4 bit descriptions

Bits	Name	Description	Reset
[63:22]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[21]	PAIRINV	Controls whether the combined result from a resource selector pair is inverted. **0b0** Do not invert the combined output of the 2 resource selectors. **0b1** Invert the combined output of the 2 resource selectors. If: • A is the register TRCRSCTLR<n>. • B is the register TRCRSCTLR<n+1>. Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows: • 0b000 -> A and B. • 0b001 -> Reserved. • 0b010 -> not(A) and B. • 0b011 -> not(A) and not(B). • 0b100 -> not(A) or not(B). • 0b101 -> not(A) or B. • 0b110 -> Reserved. • 0b111 -> A or B.	x
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR4

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0100	0b000

MSR TRCRSCTLR4, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0100	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR4

```
if 4 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[4];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[4];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[4];
```

MSR TRCRSCTLR4, <Xt>

```
if 4 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[4] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[4] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[4] = X[t, 64];
```

A.11.28 TRCRSCTLR5, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-204: AArch64_trcrsctlr5 bit assignments

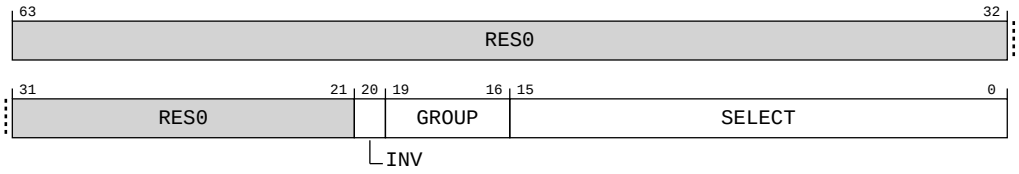


Table A-511: TRCRSCTLR5 bit descriptions

Bits	Name	Description	Reset
[63:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR5

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0101	0b000

MSR TRCRSCTLR5, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0101	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR5

```
if 5 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[5];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[5];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[5];
```

MSR TRCRSCTLR5, <Xt>

```
if 5 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[5] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[5] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[5] = X[t, 64];
```

A.11.29 TRCRSCTLR6, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-205: AArch64_trcrsctlr6 bit assignments

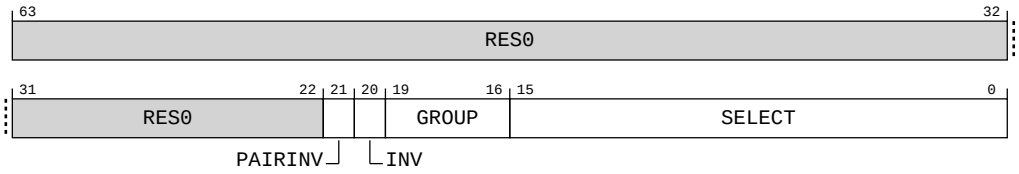


Table A-514: TRCRSCTLR6 bit descriptions

Bits	Name	Description	Reset
[63:22]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[21]	PAIRINV	Controls whether the combined result from a resource selector pair is inverted. **0b0** Do not invert the combined output of the 2 resource selectors. **0b1** Invert the combined output of the 2 resource selectors. If: • A is the register TRCRSCTLR<n>. • B is the register TRCRSCTLR<n+1>. Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows: • 0b000 -> A and B. • 0b001 -> Reserved. • 0b010 -> not(A) and B. • 0b011 -> not(A) and not(B). • 0b100 -> not(A) or not(B). • 0b101 -> not(A) or B. • 0b110 -> Reserved. • 0b111 -> A or B.	x
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR6

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0110	0b000

MSR TRCRSCTLR6, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0110	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR6

```
if 6 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[6];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[6];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[6];
```

MSR TRCRSCTLR6, <Xt>

```
if 6 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[6] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[6] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[6] = X[t, 64];
```

A.11.30 TRCRSCTLR7, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-206: AArch64_trcrsctlr7 bit assignments

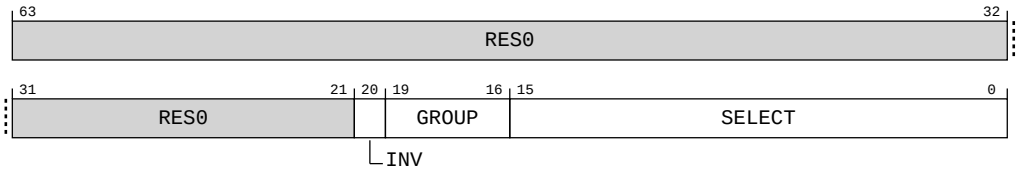


Table A-517: TRCRSCTLR7 bit descriptions

Bits	Name	Description	Reset
[63:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0111	0b000

MSR TRCRSCTLR7, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0111	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR7

```
if 7 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[7];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[7];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[7];
```

MSR TRCRSCTLR7, <Xt>

```
if 7 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[7] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[7] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[7] = X[t, 64];
```

A.11.31 TRCRSCTLR8, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-207: AArch64_trcrsctlr8 bit assignments

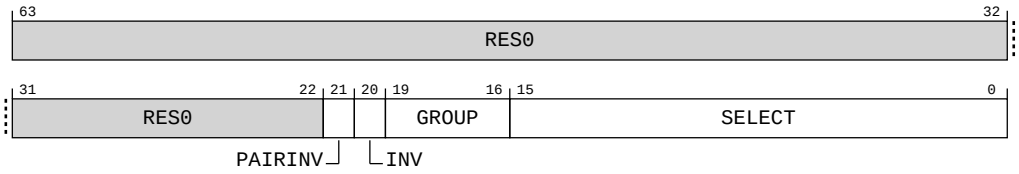


Table A-520: TRCRSCTLR8 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[21]	PAIRINV	Controls whether the combined result from a resource selector pair is inverted. **0b0** Do not invert the combined output of the 2 resource selectors. **0b1** Invert the combined output of the 2 resource selectors. If: • A is the register TRCRSCTLR<n>. • B is the register TRCRSCTLR<n+1>. Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows: • 0b000 -> A and B. • 0b001 -> Reserved. • 0b010 -> not(A) and B. • 0b011 -> not(A) and not(B). • 0b100 -> not(A) or not(B). • 0b101 -> not(A) or B. • 0b110 -> Reserved. • 0b111 -> A or B.	x
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR8

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1000	0b000

MSR TRCRSCTLR8, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1000	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR8

```
if 8 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[8];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[8];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[8];
```

MSR TRCRSCTLR8, <Xt>

```
if 8 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[8] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[8] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[8] = X[t, 64];
```

A.11.32 TRCSSCSR0, Single-shot Comparator Control Status Register <n>

Returns the status of the corresponding Single-shot Comparator Control.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

RO

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-208: AArch64_trcsscsr0 bit assignments

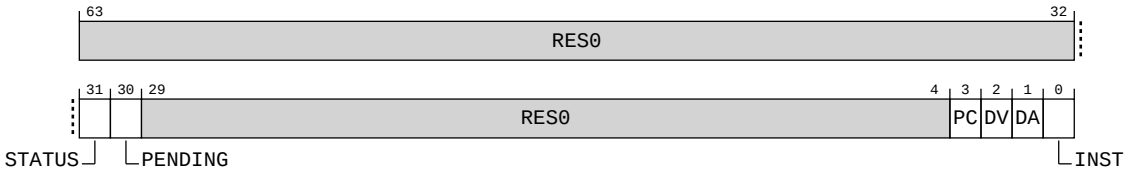


Table A-523: TRCSSCSR0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	STATUS	Single-shot Comparator Control status. Indicates if any of the comparators selected by this Single-shot Comparator control have matched. The selected comparators are defined by AArch64-TRCSSCCR<n>.ARC, AArch64-TRCSSCCR<n>.SAC, and AArch64-TRCSSPCICR<n>.PC. **0b0** No match has occurred. When the first match occurs, this field takes a value of 1. It remains at 1 until explicitly modified by a write to this register. **0b1** One or more matches has occurred. If AArch64-TRCSSCCR<n>.RST == 0 then: • There is only one match and no more matches are possible. • Software must reset this field to 0 to re-enable the Single-shot Comparator Control.	x
[30]	PENDING	Single-shot pending status. The Single-shot Comparator Control fired while the resources were in the Paused state. **0b0** No match has occurred. **0b1** One or more matches has occurred.	x
[29:4]	**RES0**	Reserved	**RES0**
[3]	PC	PE Comparator Input support. Indicates if the Single-shot Comparator Control supports PE Comparator Inputs. **0b0** This Single-shot Comparator Control does not support PE Comparator Inputs. Selecting any PE Comparator Inputs using the associated AArch64-TRCSSPCICR<n> results in **CONSTRAINED UNPREDICTABLE** behavior of the Single-shot Comparator Control resource. The Single-shot Comparator Control might match unexpectedly or might not match.	x
[2]	DV	Data value comparator support. Data value comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** This Single-shot Comparator Control does not support data value comparisons. **0b1** This Single-shot Comparator Control supports data value comparisons. This field reads as 0.	x
[1]	DA	Data Address Comparator support. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** This Single-shot Comparator Control does not support data address comparisons. **0b1** This Single-shot Comparator Control supports data address comparisons. This field reads as 0.	x

Bits	Name	Description	Reset
[0]	INST	Instruction Address Comparator support. Indicates if the Single-shot Comparator Control supports instruction address comparisons. **0b0** This Single-shot Comparator Control does not support instruction address comparisons. **0b1** This Single-shot Comparator Control supports instruction address comparisons. This field reads as 1.	x

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

MRS <Xt>, TRCSSCSR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1000	0b010

MSR TRCSSCSR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1000	0b010

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.
MRS <Xt>, TRCSSCSR0

```
if 0 >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCSR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCSR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCSSCSR[0];
```

MSR TRCSSCSR0, <Xt>

```
if 0 >= NUM_TRACE_SINGLE_SHOT_COMPARATOR_CONTROLS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSSCSR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSSCSR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCSSCSR[0] = X[t, 64];
```

A.11.33 TRCRSCTLR9, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-209: AArch64_trcrsctlr9 bit assignments

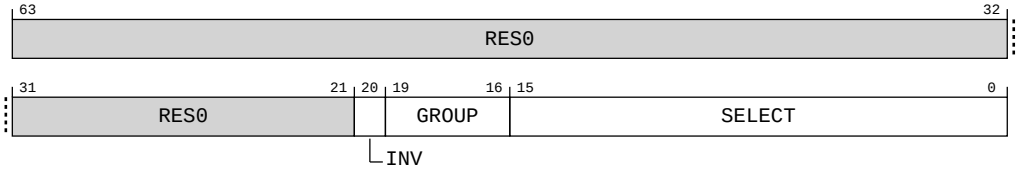


Table A-526: TRCRSCTLR9 bit descriptions

Bits	Name	Description	Reset
[63:21]	RES0	Reserved	RES0
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR9

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1001	0b000

MSR TRCRSCTLR9, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1001	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR9

```
if 9 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[9];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[9];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[9];
```

MSR TRCRSCTLR9, <Xt>

```
if 9 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[9] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[9] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[9] = X[t, 64];
```

A.11.34 TRCRSCTLR10, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-210: AArch64_trcrsctlr10 bit assignments

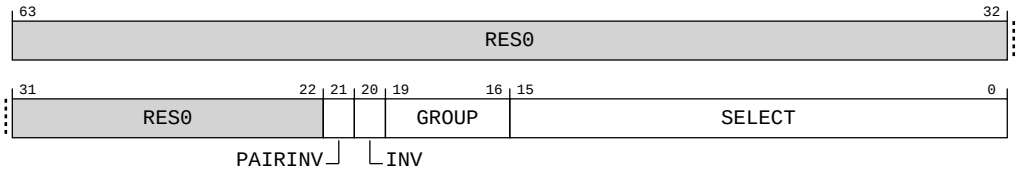


Table A-529: TRCRSCTLR10 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[21]	PAIRINV	Controls whether the combined result from a resource selector pair is inverted. **0b0** Do not invert the combined output of the 2 resource selectors. **0b1** Invert the combined output of the 2 resource selectors. If: • A is the register TRCRSCTLR<n>. • B is the register TRCRSCTLR<n+1>. Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows: • 0b000 -> A and B. • 0b001 -> Reserved. • 0b010 -> not(A) and B. • 0b011 -> not(A) and not(B). • 0b100 -> not(A) or not(B). • 0b101 -> not(A) or B. • 0b110 -> Reserved. • 0b111 -> A or B.	x
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR10

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1010	0b000

MSR TRCRSCTLR10, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1010	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR10

```
if 10 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[10];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[10];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[10];
```

MSR TRCRSCTLR10, <Xt>

```
if 10 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[10] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[10] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[10] = X[t, 64];
```

A.11.35 TRCRSCTLR11, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-211: AArch64_trcrsctlr11 bit assignments

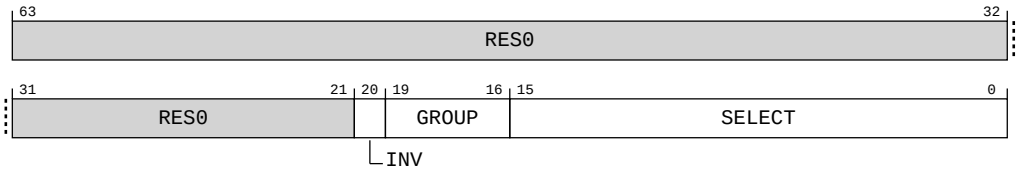


Table A-532: TRCRSCTLR11 bit descriptions

Bits	Name	Description	Reset
[63:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR11

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1011	0b000

MSR TRCRSCTLR11, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1011	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR11

```
if 11 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[11];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[11];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[11];
```

MSR TRCRSCTLR11, <Xt>

```
if 11 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[11] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[11] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[11] = X[t, 64];
```

A.11.36 TRCRSCTLR12, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-212: AArch64_trcrsctlr12 bit assignments

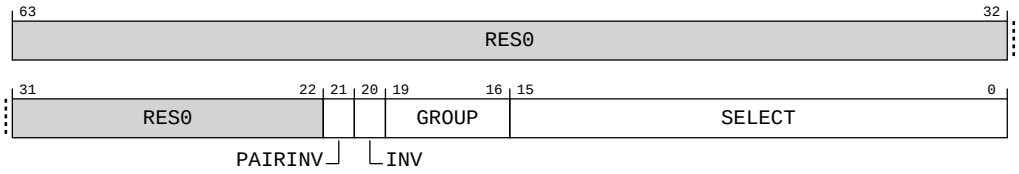


Table A-535: TRCRSCTLR12 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[21]	PAIRINV	Controls whether the combined result from a resource selector pair is inverted. **0b0** Do not invert the combined output of the 2 resource selectors. **0b1** Invert the combined output of the 2 resource selectors. If: • A is the register TRCRSCTLR<n>. • B is the register TRCRSCTLR<n+1>. Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows: • 0b000 -> A and B. • 0b001 -> Reserved. • 0b010 -> not(A) and B. • 0b011 -> not(A) and not(B). • 0b100 -> not(A) or not(B). • 0b101 -> not(A) or B. • 0b110 -> Reserved. • 0b111 -> A or B.	x
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR12

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1100	0b000

MSR TRCRSCTLR12, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1100	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR12

```
if 12 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[12];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[12];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[12];
```

MSR TRCRSCTLR12, <Xt>

```
if 12 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[12] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[12] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[12] = X[t, 64];
```

A.11.37 TRCRSCTLR13, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-213: AArch64_trcrsctlr13 bit assignments

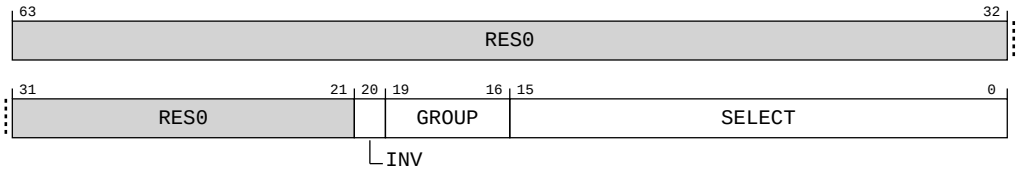


Table A-538: TRCRSCTLR13 bit descriptions

Bits	Name	Description	Reset
[63:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR13

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1101	0b000

MSR TRCRSCTLR13, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1101	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR13

```
if 13 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[13];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[13];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[13];
```

MSR TRCRSCTLR13, <Xt>

```
if 13 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[13] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[13] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[13] = X[t, 64];
```

A.11.38 TRCRSCTLR14, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-214: AArch64_trcrsctlr14 bit assignments

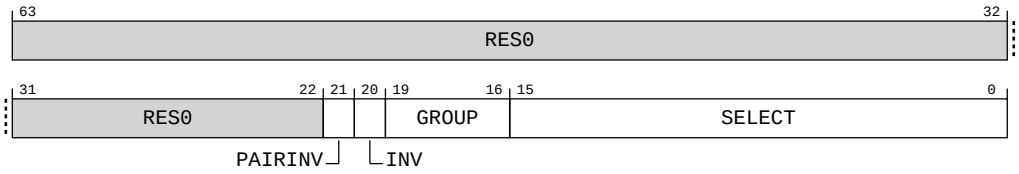


Table A-541: TRCRSCTLR14 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[21]	PAIRINV	Controls whether the combined result from a resource selector pair is inverted. **0b0** Do not invert the combined output of the 2 resource selectors. **0b1** Invert the combined output of the 2 resource selectors. If: • A is the register TRCRSCTLR<n>. • B is the register TRCRSCTLR<n+1>. Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows: • 0b000 -> A and B. • 0b001 -> Reserved. • 0b010 -> not(A) and B. • 0b011 -> not(A) and not(B). • 0b100 -> not(A) or not(B). • 0b101 -> not(A) or B. • 0b110 -> Reserved. • 0b111 -> A or B.	x
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x

Bits	Name	Description	Reset
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR14

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1110	0b000

MSR TRCRSCTLR14, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1110	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR14

```
if 14 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[14];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[14];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[14];
```

MSR TRCRSCTLR14, <Xt>

```
if 14 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[14] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[14] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[14] = X[t, 64];
```

A.11.39 TRCRSCTLR15, Resource Selection Control Register <n>

Controls the selection of the resources in the trace unit.

Configurations

Resource selector 0 always returns FALSE.

Resource selector 1 always returns TRUE.

Resource selectors are implemented in pairs. Each odd numbered resource selector is part of a pair with the even numbered resource selector that is numbered as one less than it. For example, resource selectors 2 and 3 form a pair.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-215: AArch64_trcrsctlr15 bit assignments

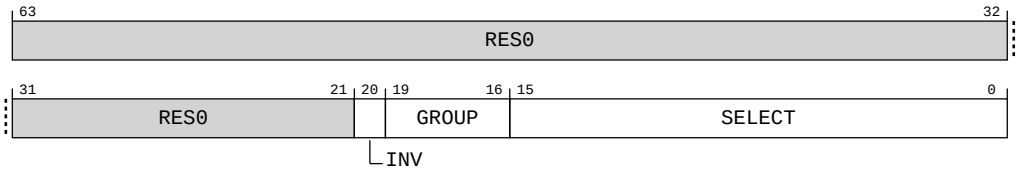


Table A-544: TRCRSCTLR15 bit descriptions

Bits	Name	Description	Reset
[63:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	INV	Controls whether the resource, that TRCRSCTLR<n>.GROUP and TRCRSCTLR<n>.SELECT selects, is inverted. **0b0** Do not invert the output of this selector. **0b1** Invert the output of this selector.	x
[19:16]	GROUP	Selects a group of resources. **0b0000** External Input Selectors. **0b0001** PE Comparator Inputs. **0b0010** Counters and Sequencer. **0b0011** Single-shot Comparator Controls. **0b0100** Single Address Comparators. **0b0101** Address Range Comparators. **0b0110** Context Identifier Comparators. **0b0111** Virtual Context Identifier Comparators. All other values are reserved.	xxxx

Bits	Name	Description	Reset
15:0	SELECT	**SELECT encoding for External Input Selectors** **15:4** Reserved, **RES0**. **EXTIN[<m>], bit[m], for m = 3 to 0** Selects one or more External Inputs. **0b0** Ignore EXTIN <m>. **0b1** Select EXTIN <m>. **SELECT encoding for PE Comparator Inputs** **15:8** Reserved, **RES0**. **PECOMP[<m>], bit[m], for m = 7 to 0** Selects one or more PE Comparator Inputs. **0b0** Ignore PE Comparator Input <m>. **0b1** Select PE Comparator Input <m>. **SELECT encoding for Counters and Sequencer** **15:8** Reserved, **RES0**. **SEQUENCER[<m>], bit[m], for m = 3 to 0** Sequencer states. **0b0** Ignore Sequencer state <m>. **0b1** Select Sequencer state <m>. **COUNTERS[<m>], bit[m], for m = 3 to 0** Counters resources at zero. **0b0** Ignore Counter <m>. **0b1** Select Counter <m> is zero. **SELECT encoding for Single-shot Comparator Controls** **15:8** Reserved, **RES0**. **SINGLE_SHOT[<m>], bit[m], for m = 7 to 0** Selects one or more Single-shot Comparator Controls. **0b0** Ignore Single-shot Comparator Control <m>. **0b1** Select Single-shot Comparator Control <m>. **SELECT encoding for Single Address Comparators**	16{x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR15

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1111	0b000

MSR TRCRSCTLR15, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b1111	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT0.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT1.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT2.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 0 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n.
- AArch64-TRCEVENTCTL0R.EVENT3.TYPE == 1 and AArch64-TRCEVENTCTL0R.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 0 and AArch64-TRCSEQRSTEVR.RST.SEL == n.
- AArch64-TRCSEQRSTEVR.RST.TYPE == 1 and AArch64-TRCSEQRSTEVR.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

MRS <Xt>, TRCRSCTLR15

```
if 15 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[15];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[15];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCRSCTLR[15];
```

MSR TRCRSCTLR15, <Xt>

```
if 15 >= NUM_TRACE_RESOURCE_SELECTOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[15] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCRSCTLR[15] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

```
        TRCRSCTLR[15] = X[t, 64];
```

A.11.40 TRCACVR0, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-216: AArch64_trcacvr0 bit assignments

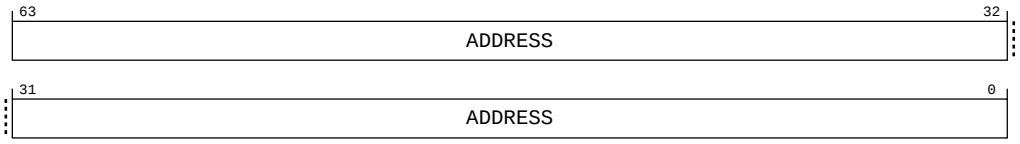


Table A-547: TRCACVR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR0

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0000`	`0b000`

MSR TRCACVR0, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0000`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR0

```
if 0 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[0];
```

MSR TRCACVR0, <Xt>

```
if 0 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[0] = X[t, 64];
```

A.11.41 TRCACATR0, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-217: AArch64_trcacatr0 bit assignments

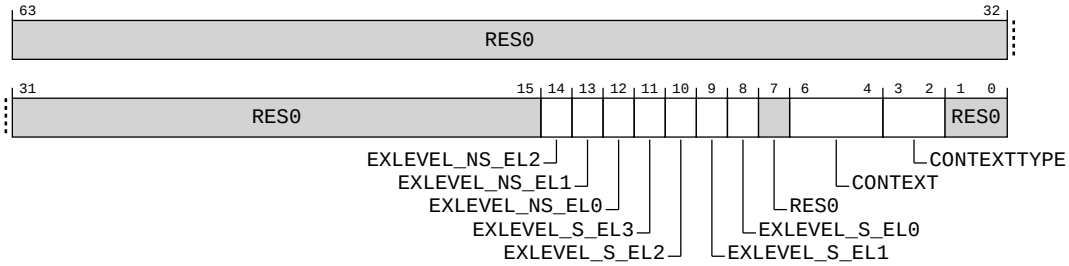


Table A-550: TRCACATR0 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0000	0b010

MSR TRCACATR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0000	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR0

```
if 0 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[0];
```

MSR TRCACATR0, <Xt>

```
if 0 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[0] = X[t, 64];
```

A.11.42 TRCACVR1, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-218: AArch64_trcacvr1 bit assignments

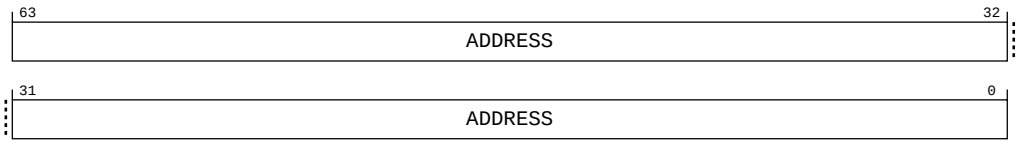


Table A-553: TRCACVR1 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR1

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0010`	`0b000`

MSR TRCACVR1, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0010`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR1

```
if 1 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[1];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[1];
```

MSR TRCACVR1, <Xt>

```
if 1 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[1] = X[t, 64];
```

A.11.43 TRCACATR1, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-219: AArch64_trcacatr1 bit assignments

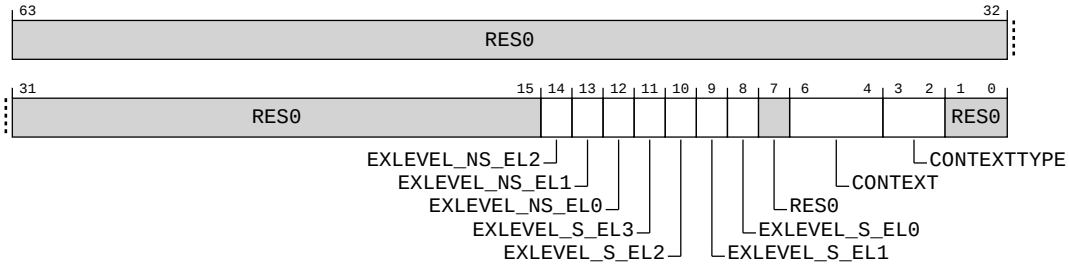


Table A-556: TRCACATR1 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0010	0b010

MSR TRCACATR1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0010	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR1

```
if 1 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[1];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[1];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[1];
```

MSR TRCACATR1, <Xt>

```
if 1 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[1] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[1] = X[t, 64];
```

A.11.44 TRCACVR2, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-220: AArch64_trcacvr2 bit assignments

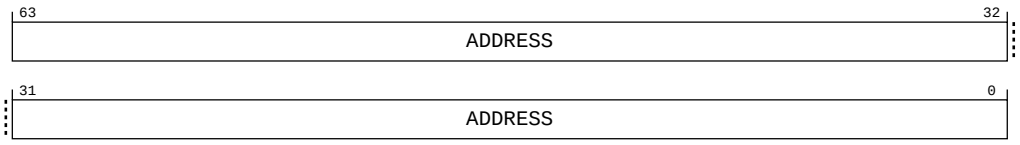


Table A-559: TRCACVR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR2

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0100`	`0b000`

MSR TRCACVR2, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0100`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR2

```
if 2 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[2];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[2];
```

MSR TRCACVR2, <Xt>

```
if 2 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[2] = X[t, 64];
```

A.11.45 TRCACATR2, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-221: AArch64_trcacatr2 bit assignments

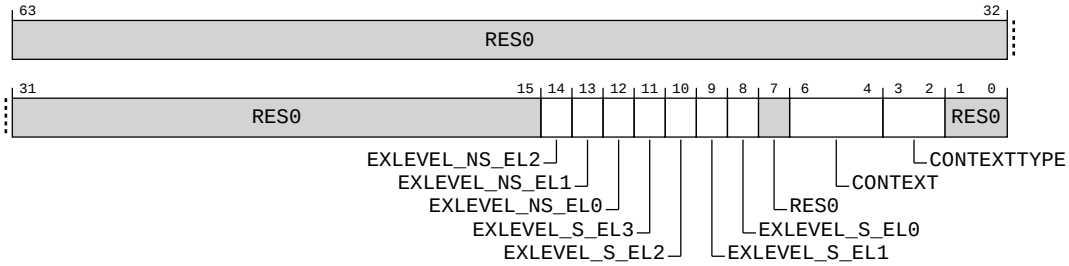


Table A-562: TRCACATR2 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR2

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0100	0b010

MSR TRCACATR2, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0100	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR2

```
if 2 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[2];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[2];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[2];
```

MSR TRCACATR2, <Xt>

```
if 2 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[2] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[2] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[2] = X[t, 64];
```

A.11.46 TRCACVR3, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-222: AArch64_trcacvr3 bit assignments

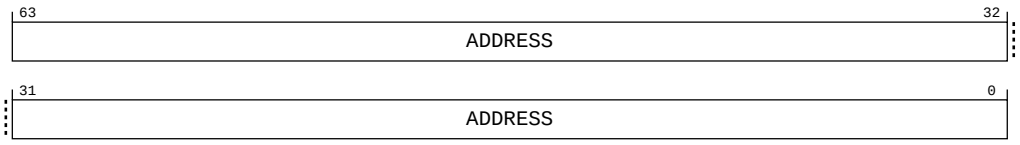


Table A-565: TRCACVR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR3

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0110`	`0b000`

MSR TRCACVR3, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b0110`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR3

```
if 3 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[3];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[3];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[3];
```

MSR TRCACVR3, <Xt>

```
if 3 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[3] = X[t, 64];
```

A.11.47 TRCACATR3, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-223: AArch64_trcacatr3 bit assignments

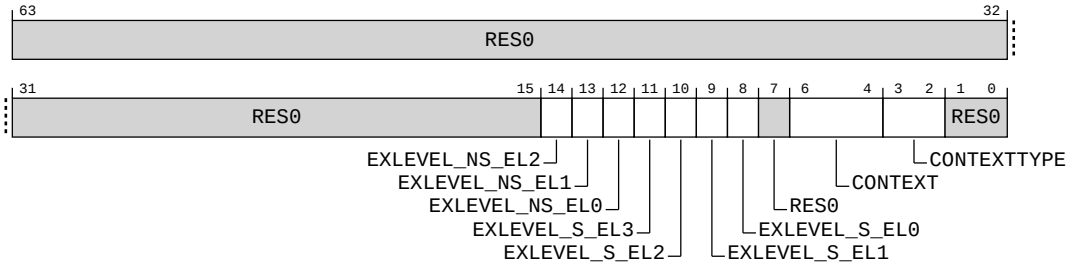


Table A-568: TRCACATR3 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR3

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0110	0b010

MSR TRCACATR3, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b0110	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR3

```
if 3 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[3];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[3];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[3];
```

MSR TRCACATR3, <Xt>

```
if 3 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[3] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[3] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[3] = X[t, 64];
```

A.11.48 TRCACVR4, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-224: AArch64_trcacvr4 bit assignments

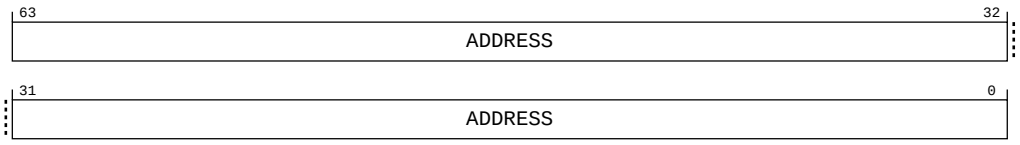


Table A-571: TRCACVR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR4

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1000`	`0b000`

MSR TRCACVR4, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1000`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR4

```
if 4 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[4];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[4];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[4];
```

MSR TRCACVR4, <Xt>

```
if 4 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[4] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[4] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[4] = X[t, 64];
```

A.11.49 TRCACATR4, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-225: AArch64_trcacatr4 bit assignments

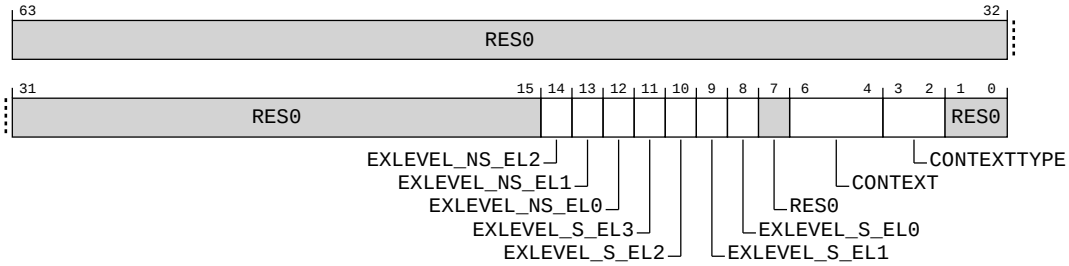


Table A-574: TRCACATR4 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR4

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1000	0b010

MSR TRCACATR4, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1000	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR4

```
if 4 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[4];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[4];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[4];
```

MSR TRCACATR4, <Xt>

```
if 4 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[4] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[4] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[4] = X[t, 64];
```

A.11.50 TRCACVR5, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-226: AArch64_trcacvr5 bit assignments

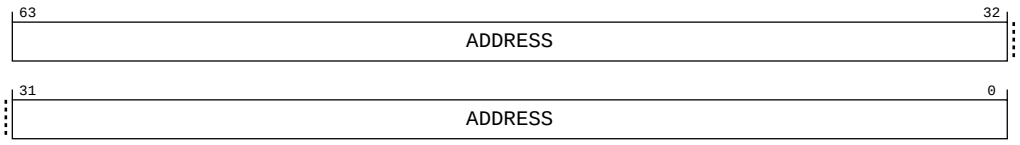


Table A-577: TRCACVR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR5

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1010`	`0b000`

MSR TRCACVR5, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1010`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR5

```
if 5 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[5];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[5];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[5];
```

MSR TRCACVR5, <Xt>

```
if 5 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[5] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[5] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[5] = X[t, 64];
```

A.11.51 TRCACATR5, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-227: AArch64_trcacatr5 bit assignments

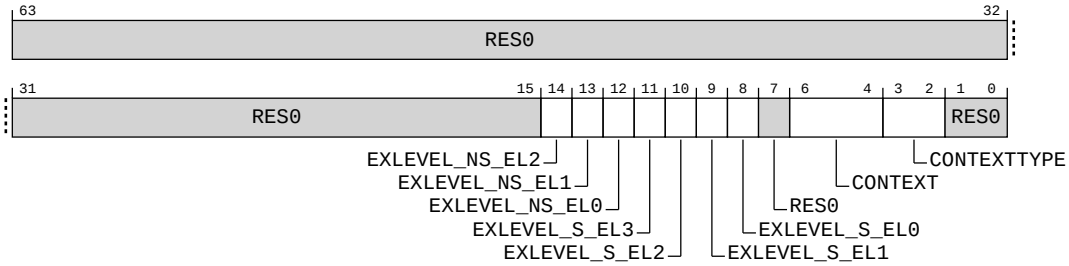


Table A-580: TRCACATR5 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR5

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1010	0b010

MSR TRCACATR5, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1010	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR5

```
if 5 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[5];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[5];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[5];
```

MSR TRCACATR5, <Xt>

```
if 5 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[5] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[5] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[5] = X[t, 64];
```

A.11.52 TRCACVR6, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-228: AArch64_trcacvr6 bit assignments

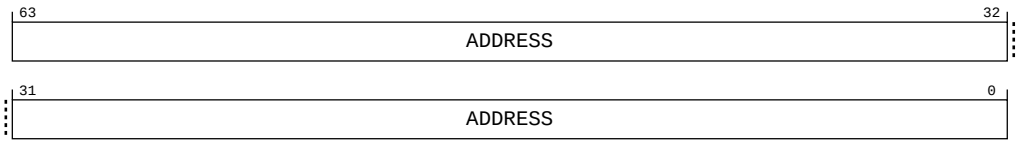


Table A-583: TRCACVR6 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR6

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1100`	`0b000`

MSR TRCACVR6, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1100`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR6

```
if 6 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[6];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[6];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[6];
```

MSR TRCACVR6, <Xt>

```
if 6 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[6] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[6] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[6] = X[t, 64];
```

A.11.53 TRCACATR6, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-229: AArch64_trcacatr6 bit assignments

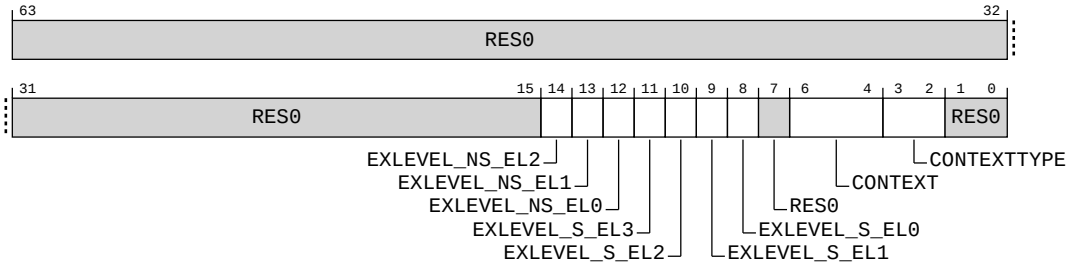


Table A-586: TRCACATR6 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR6

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1100	0b010

MSR TRCACATR6, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1100	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR6

```
if 6 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[6];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[6];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[6];
```

MSR TRCACATR6, <Xt>

```
if 6 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[6] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[6] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[6] = X[t, 64];
```

A.11.54 TRCACVR7, Address Comparator Value Register <n>

Contains the address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-230: AArch64_trcacvr7 bit assignments

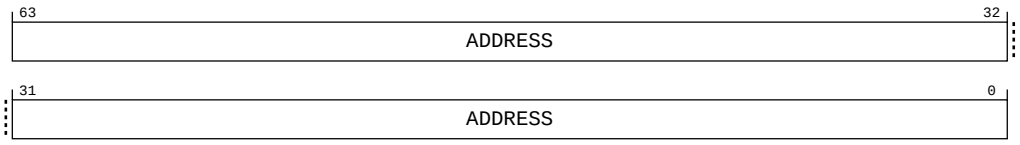


Table A-589: TRCACVR7 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Address Value. The Address Comparators can support implementations that use multiple address widths. When the trace unit compares the ADDRESS field with an address that has a width less than this field, then the address must be zero-extended to the ADDRESS field width. The trace unit then compares all implemented bits. For example, in a system that supports both 32-bit and 64-bit addresses, when the PE is in AArch32 state the comparator must zero-extend the 32-bit address and compare against the full 64 bits that are stored in the TRCACVR<n>. This requires that the trace analyzer always programs all implemented bits of the TRCACVR<n>. The result of writing a value other than all zeros or all ones to ADDRESS at bits[63:P] is an **UNKNOWN** value, where P is defined as the maximum virtual address size supported by the PE. The result of writing a value of all zeros or all ones to ADDRESS at bits[63:P] is the written value, and a read of the register returns the written value.	`64{x}`

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == `0b0100` and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == `0b0101` and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACVR7

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1110`	`0b000`

MSR TRCACVR7, <Xt>

op0	op1	CRn	CRm	op2
`0b10`	`0b001`	`0b0010`	`0b1110`	`0b000`

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACVR7

```
if 7 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[7];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[7];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACVR[7];
```

MSR TRCACVR7, <Xt>

```
if 7 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[7] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[7] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACVR[7] = X[t, 64];
```

A.11.55 TRCACATR7, Address Comparator Access Type Register <n>

Defines the type of access for the corresponding AArch64-TRCACVR<n> Register. This register configures the context type, Exception levels, alignment, masking that is applied by the Address Comparator, and how the Address Comparator behaves when it is one half of an Address Range Comparator.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-231: AArch64_trcacatr7 bit assignments

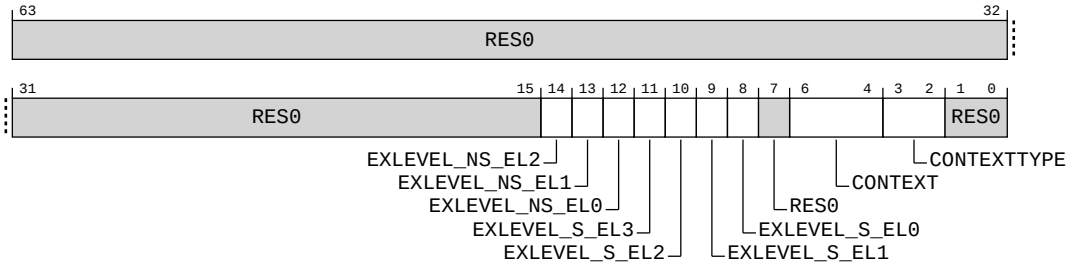


Table A-592: TRCACATR7 bit descriptions

Bits	Name	Description	Reset
[63:15]	**RES0**	Reserved	**RES0**
[14]	EXLEVEL_NS_EL2	Non-secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL2. **0b1** The Address Comparator does not perform comparisons in Non-secure EL2.	x
[13]	EXLEVEL_NS_EL1	Non-secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL1. **0b1** The Address Comparator does not perform comparisons in Non-secure EL1.	x
[12]	EXLEVEL_NS_EL0	Non-secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Non-secure state. **0b0** The Address Comparator performs comparisons in Non-secure EL0. **0b1** The Address Comparator does not perform comparisons in Non-secure EL0.	x
[11]	EXLEVEL_S_EL3	EL3 address comparison control. Controls whether a comparison can occur at EL3. **0b0** The Address Comparator performs comparisons in EL3. **0b1** The Address Comparator does not perform comparisons in EL3.	x
[10]	EXLEVEL_S_EL2	Secure EL2 address comparison control. Controls whether a comparison can occur at EL2 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL2. **0b1** The Address Comparator does not perform comparisons in Secure EL2.	x

Bits	Name	Description	Reset
[9]	EXLEVEL_S_EL1	Secure EL1 address comparison control. Controls whether a comparison can occur at EL1 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL1. **0b1** The Address Comparator does not perform comparisons in Secure EL1.	x
[8]	EXLEVEL_S_EL0	Secure EL0 address comparison control. Controls whether a comparison can occur at EL0 in Secure state. **0b0** The Address Comparator performs comparisons in Secure EL0. **0b1** The Address Comparator does not perform comparisons in Secure EL0.	x
[7]	**RES0**	Reserved	**RES0**
[6:4]	CONTEXT	Selects a Context Identifier Comparator or Virtual Context Identifier Comparator: **0b000** Comparator 0. The width of this field is dependent on the maximum number of Context Identifier Comparators or Virtual Context Identifier Comparators implemented. Unimplemented bits are **RES0**.	xxx
[3:2]	CONTEXTTYPE	Controls whether the Address Comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons. **0b00** The Address Comparator is not dependent on the Context Identifier Comparators or Virtual Context Identifier Comparators. **0b01** The Address Comparator is dependent on the Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Context Identifier Comparator and the address comparison match. **0b10** The Address Comparator is dependent on the Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if both the Virtual Context Identifier Comparator and the address comparison match. **0b11** The Address Comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator that TRCACATR<n>.CONTEXT specifies. The Address Comparator signals a match only if the Context Identifier Comparator, the Virtual Context Identifier Comparator, and address comparison all match. If AArch64-TRCIDR4.NUMCIDC == 0b0000, then bit [2] is **RES0**. If AArch64-TRCIDR4.NUMVMIDC == 0b0000, then bit [3] is **RES0**.	xx
[1:0]	**RES0**	Reserved	**RES0**

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.

- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCACATR7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1110	0b010

MSR TRCACATR7, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	0b1110	0b010

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 1.
- AArch64-TRCVISSCTLR.START[n] == 1.
- AArch64-TRCVISSCTLR.STOP[n] == 1.
- TRCSSCCR<>.ARC[n/2] == 1.
- TRCSSCCR<>.SAC[n] == 1.
- AArch64-TRCQCTLR.RANGE[n/2] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCACATR7

```
if 7 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[7];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[7];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCACATR[7];
```

MSR TRCACATR7, <Xt>

```
if 7 >= NUM_TRACE_ADDRESS_COMPARATOR_PAIRS * 2 then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[7] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[7] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCACATR[7] = X[t, 64];
```

A.11.56 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 unit registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-232: AArch64_trccidcvr0 bit assignments

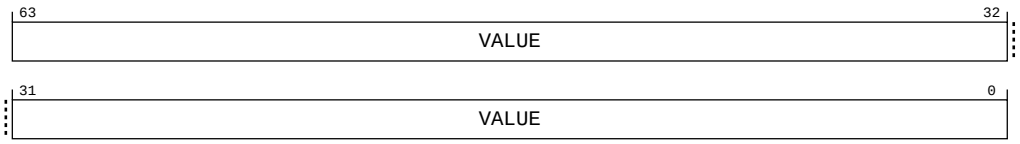


Table A-595: TRCCIDCVR0 bit descriptions

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.	`64{x}`

Access

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == `0b0110` and TRCRSCTLR<a>.CID[n] == 1.
- TRCACATR<a>.CONTEXTTYPE == `0b01` or `0b11` and TRCACATR<a>.CONTEXT == n.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

MRS <Xt>, TRCCIDCVR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b000

MSR TRCCIDCVR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0110 and TRCRSCTLR<a>.CID[n] == 1.
- TRCACATR<a>.CONTEXTTYPE == 0b01 or 0b11 and TRCACATR<a>.CONTEXT == n.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.
MRS <Xt>, TRCCIDCVR0

```
if 0 >= NUM_TRACE_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCVR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCVR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCIDCVR[0];
```

MSR TRCCIDCVR0, <Xt>

```
if 0 >= NUM_TRACE_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCVR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCVR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCIDCVR[0] = X[t, 64];
```

A.11.57 TRCVMIDCVR0, Virtual Context Identifier Comparator Value Register <n>

Contains the Virtual Context Identifier Comparator value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-233: AArch64_trcvmidcvr0 bit assignments

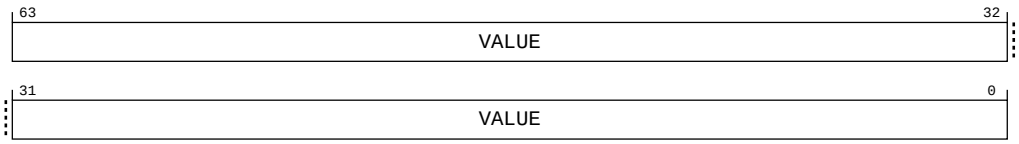


Table A-598: TRCVMIDCVR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	VALUE	Virtual context identifier value. The width of this field is indicated by AArch64-TRCIDR2.VMIDSIZE. Unimplemented bits are **RES0**. After a PE Reset, the trace unit assumes that the Virtual context identifier is zero until the PE updates the Virtual context identifier .	64{x}

Access

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0111 and TRCRSCTLR<a>.VMID[n] == 1.
- TRCACATR<a>.CONTEXTTYPE == 0b10 or 0b11 and TRCACATR<a>.CONTEXT == n.

MRS <Xt>, TRCVMIDCVR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b001

MSR TRCVMIDCVR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b001

Accessibility

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0111 and TRCRSCTLR<a>.VMID[n] == 1.
- TRCACATR<a>.CONTEXTTYPE == 0b10 or 0b11 and TRCACATR<a>.CONTEXT == n.

MRS <Xt>, TRCVMIDCVR0

```
if 0 >= NUM_TRACE_VIRTUAL_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVMIDCVR[0];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVMIDCVR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVMIDCVR[0];
```

MSR TRCVMIDCVR0, <Xt>

```
if 0 >= NUM_TRACE_VIRTUAL_CONTEXT_IDENTIFIER_COMPARATORS then
    UNDEFINED;
elsif 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 CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVMIDCVR[0] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVMIDCVR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVMIDCVR[0] = X[t, 64];
```

A.12 AArch64 Memory Partitioning and Monitoring registers summary

The summary table provides an overview of all Memory Partitioning and Monitoring registers in the core. For more information about a register, click the register name in the table.

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

Table A-601: Memory Partitioning and Monitoring registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MPAM1_EL1	3	0	C10	C5	0	–	64-bit	MPAM1 Register (EL1)
MPAM0_EL1	3	0	C10	C5	1	–	64-bit	MPAM0 Register (EL1)
MPAMHCR_EL2	3	4	C10	C4	0	–	64-bit	MPAM Hypervisor Control Register (EL2)
MPAMVPMV_EL2	3	4	C10	C4	1	–	64-bit	MPAM Virtual Partition Mapping Valid Register
MPAM2_EL2	3	4	C10	C5	0	–	64-bit	MPAM2 Register (EL2)
MPAMVPM0_EL2	3	4	C10	C6	0	–	64-bit	MPAM Virtual PARTID Mapping Register 0
MPAMVPM1_EL2	3	4	C10	C6	1	–	64-bit	MPAM Virtual PARTID Mapping Register 1
MPAM3_EL3	3	6	C10	C5	0	–	64-bit	MPAM3 Register (EL3)

A.12.1 MPAMVPMV_EL2, MPAM Virtual Partition Mapping Valid Register

Valid bits for virtual PARTID mapping entries. Each bit m corresponds to virtual PARTID mapping entry m in the MPAMVPM<n>_EL2 registers where n = m >> 2.

Configurations

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

Attributes

Width

64

Functional group

Memory Partitioning and Monitoring registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-234: AArch64_mpamvpmv_el2 bit assignments

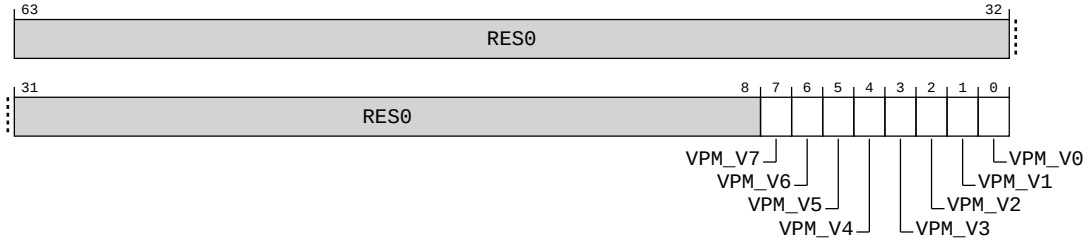


Table A-602: MPAMVPMV_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7]	VPM_V7	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[6]	VPM_V6	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[5]	VPM_V5	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[4]	VPM_V4	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[3]	VPM_V3	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[2]	VPM_V2	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[1]	VPM_V1	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x
[0]	VPM_V0	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	x

Access

MRS <Xt>, MPAMVPMV_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

MSR MPAMVPMV_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

Accessibility

MRS <Xt>, MPAMVPMV_EL2

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

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPMV_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPMV_EL2;
```

MSR MPAMVPMV_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPMV_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPMV_EL2 = X[t, 64];
```

A.12.2 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 valid only when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

Memory Partitioning and Monitoring registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-235: AArch64_mpamvpm0_el2 bit assignments

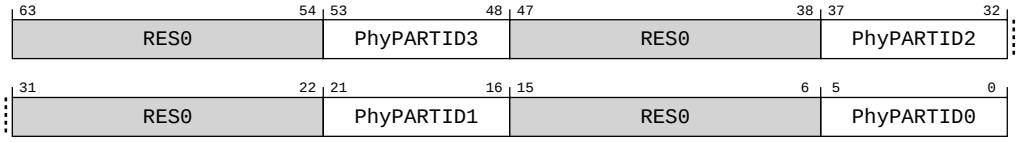


Table A-605: MPAMVPM0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:54]	**RES0**	Reserved	**RES0**
[53:48]	PhyPARTID3	Virtual PARTID Mapping Entry for virtual PARTID 3. PhyPARTID3 gives the mapping of virtual PARTID 3 to a physical PARTID.	6{x}
[47:38]	**RES0**	Reserved	**RES0**
[37:32]	PhyPARTID2	Virtual PARTID Mapping Entry for virtual PARTID 2. PhyPARTID2 gives the mapping of virtual PARTID 2 to a physical PARTID.	6{x}
[31:22]	**RES0**	Reserved	**RES0**
[21:16]	PhyPARTID1	Virtual PARTID Mapping Entry for virtual PARTID 1. PhyPARTID1 gives the mapping of virtual PARTID 1 to a physical PARTID.	6{x}
[15:6]	**RES0**	Reserved	**RES0**
[5:0]	PhyPARTID0	Virtual PARTID Mapping Entry for virtual PARTID 0. PhyPARTID0 gives the mapping of virtual PARTID 0 to a physical PARTID.	6{x}

Access

MRS <Xt>, MPAMVPM0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

MSR MPAMVPM0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

Accessibility

MRS <Xt>, MPAMVPM0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM0_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM0_EL2;
```

MSR MPAMVPM0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM0_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM0_EL2 = X[t, 64];
```

A.12.3 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 valid only when the AArch64-MPAMVPMV_EL2.VPM_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

Memory Partitioning and Monitoring registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-236: AArch64_mpamvpm1_el2 bit assignments

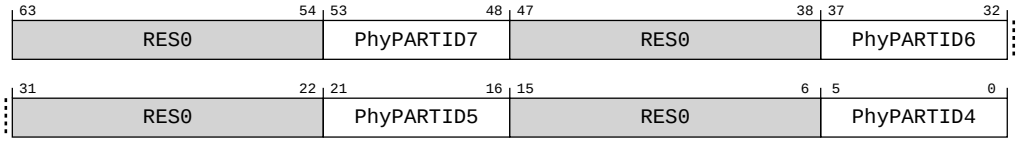


Table A-608: MPAMVPM1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:54]	**RES0**	Reserved	**RES0**
[53:48]	PhyPARTID7	Virtual PARTID Mapping Entry for virtual PARTID 7. PhyPARTID7 gives the mapping of virtual PARTID 7 to a physical PARTID.	6{x}
[47:38]	**RES0**	Reserved	**RES0**
[37:32]	PhyPARTID6	Virtual PARTID Mapping Entry for virtual PARTID 6. PhyPARTID6 gives the mapping of virtual PARTID 6 to a physical PARTID.	6{x}
[31:22]	**RES0**	Reserved	**RES0**
[21:16]	PhyPARTID5	Virtual PARTID Mapping Entry for virtual PARTID 5. PhyPARTID5 gives the mapping of virtual PARTID 5 to a physical PARTID.	6{x}
[15:6]	**RES0**	Reserved	**RES0**
[5:0]	PhyPARTID4	Virtual PARTID Mapping Entry for virtual PARTID 4. PhyPARTID4 gives the mapping of virtual PARTID 4 to a physical PARTID.	6{x}

Access

MRS <Xt>, MPAMVPM1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

MSR MPAMVPM1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

Accessibility

MRS <Xt>, MPAMVPM1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM1_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM1_EL2;
```

MSR MPAMVPM1_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM1_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM1_EL2 = X[t, 64];
```

A.13 AArch64 RAS registers summary

The summary table provides an overview of all RAS registers in the core. For more information about a register, click the register name in the table.

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

Table A-611: RAS registers summary

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

A.13.1 ERRIDR_EL1, Error Record ID Register

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-237: AArch64_erridr_el1 bit assignments

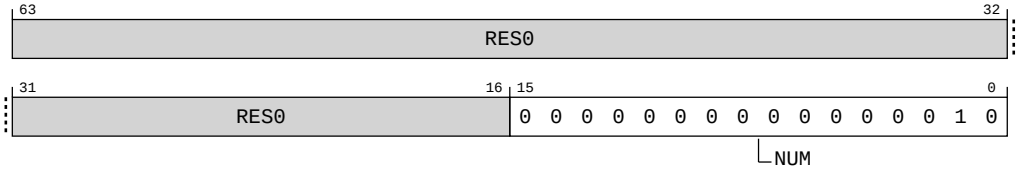


Table A-612: ERRIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:0]	NUM	Highest numbered index of the records that can be accessed through the Error Record System registers plus one. Zero indicates no records can be accessed through the Error Record System registers. Each implemented record is owned by a node. A node might own multiple records. **0b0000000000000010** Two Records Present.	0x0002

Access

MRS <Xt>, ERRIDR_EL1

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

Accessibility

MRS <Xt>, ERRIDR_EL1

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

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERRIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERRIDR_EL1;
```

A.13.2 ERRSELR_EL1, Error Record Select Register

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

Configurations

If AArch64-ERRIDR_EL1 indicates that zero error records are implemented, then it is IMPLEMENTATION DEFINED whether ERRSELR_EL1 is UNDEFINED or RES0.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-238: AArch64_errselr_el1 bit assignments

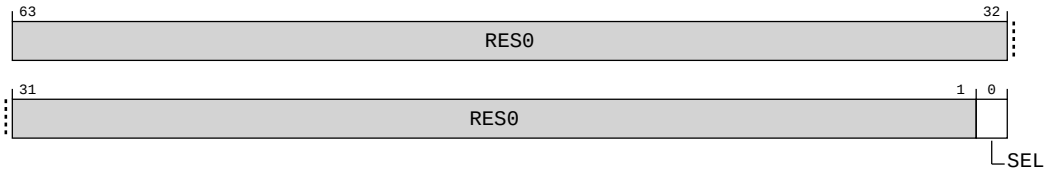


Table A-614: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	SEL	**0b0** Selects record 0, containing errors from DSU RAMs **0b1** Selects record 1, containing errors from Core RAMs	0b0

Access

MRS <Xt>, ERRSELR_EL1

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

MSR ERRSELR_EL1, <Xt>

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

Accessibility

MRS <Xt>, ERRSELR_EL1

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

MSR ERRSELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
```

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

A.13.3 ERXFR_EL1, Selected Error Record Feature Register

Accesses ext-ERR<n>FR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0001 0000 0010 1001 1010 0010

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-239: AArch64_erxfr_el1 bit assignments

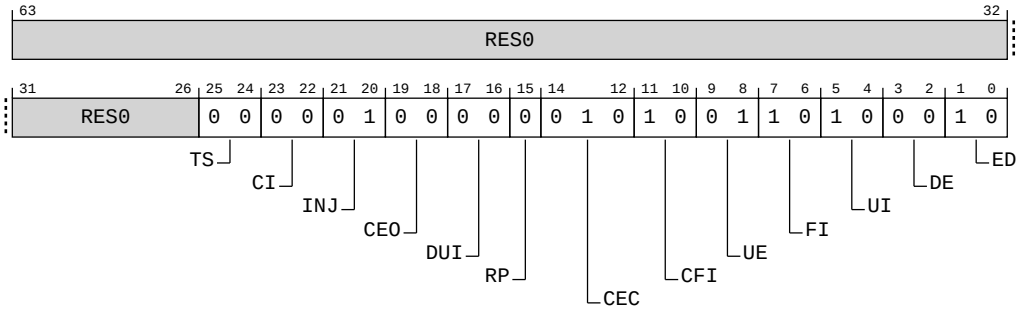


Table A-617: ERXFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:26]	**RES0**	Reserved	**RES0**
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, ERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** The node does not support a timestamp register.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. **0b00** Does not support the critical error interrupt. ERXCTLR_EL1.CI is **RES0**.	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. **0b01** The node implements the RAS Common Fault Injection Model Extension. See ERXPFGF_EL1 for more information.	0b01
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior when a second Corrected error is detected after a first Corrected error has been recorded by an error record <m> owned by the node. **0b00** Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then ERXSTATUS_EL1.OF is set to 0b1.	0b00
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the control for enabling error recovery interrupts on deferred errors are implemented. **0b00** Does not support the control for enabling error recovery interrupts on deferred errors. ERXCTLR_EL1.DUI is **RES0**.	0b00
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. **0b01** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	0b0

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].	0b010
[11:10]	CFI	Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented. **0b10** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI.	0b10
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. **0b01** In-band uncorrected error reporting (External Aborts) is supported and always enabled. ERXCTLR_EL1.UE is **RES0**.	0b01
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. **0b10** Fault handling interrupt is supported and controllable using ERXCTLR_EL1.FI.	0b10
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. **0b10** Error handling interrupt is supported and controllable using ERXCTLR_EL1.UI.	0b10
[3:2]	DE	**0b00**	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.	0b10

Access

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

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

MRS <Xt>, ERXFR_EL1

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

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

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

MRS <Xt>, ERXFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXFR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXFR_EL1;
```

A.13.4 ERXCTLR_EL1, Selected Error Record Control Register

Accesses ext-ERR<n>CTLR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 xxxx 00x0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-240: AArch64_erxctlr_el1 bit assignments

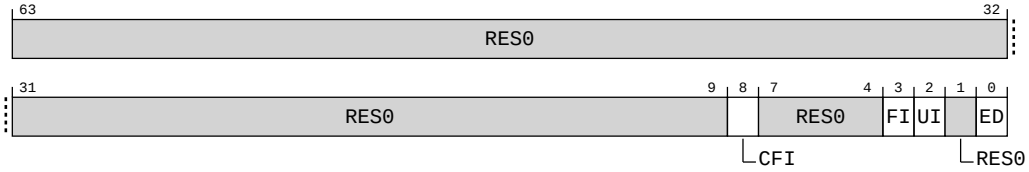


Table A-619: ERXCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:9]	**RES0**	Reserved	**RES0**
[8]	CFI	Fault handling interrupt for Corrected errors enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated when one of the standard CE counters on ERXMISC0_EL1 overflows and the overflow bit is set. The possible values are: **0b0** Fault handling interrupt not generated for Corrected errors. **0b1** Fault handling interrupt generated for Corrected errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error. Cold reset only. Unaffected by Warm reset	0b0
[7:4]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[3]	FI	Fault handling interrupt enable. This control applies to errors arising from both reads and writes. The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are: **0b0** Fault handling interrupt disabled. **0b1** Fault handling interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error. Cold reset only. Unaffected by Warm reset	0b0
[2]	UI	Uncorrected error recovery interrupt enable. This control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. **0b0** Error recovery interrupt disabled. **0b1** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error. Cold reset only. Unaffected by Warm reset	0b0
[1]	**RES0**	Reserved	**RES0**
[0]	ED	Error Detection and correction enable. The possible values are: **0b0** Error detection and correction disabled. **0b1** Error detection and correction enabled. Cold reset only. Unaffected by Warm reset	0b0

Access

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

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

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>CTLR is not present, meaning reads and writes of ERXCTLR_EL1 are **RES0**.

MRS <Xt>, ERXCTLR_EL1

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

MSR ERXCTLR_EL1, <Xt>

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

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

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

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>CTLR is not present, meaning reads and writes of ERXCTLR_EL1 are RES0.
MRS <Xt>, ERXCTLR_EL1

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

MSR ERXCTLR_EL1, <Xt>

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

```
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXCTLR_EL1 = X[t, 64];
```

A.13.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register

Accesses ext-ERR<n>STATUS for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 xxxx xxxx xxxx xxx0 0000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-241: AArch64_erxstatus_el1 bit assignments

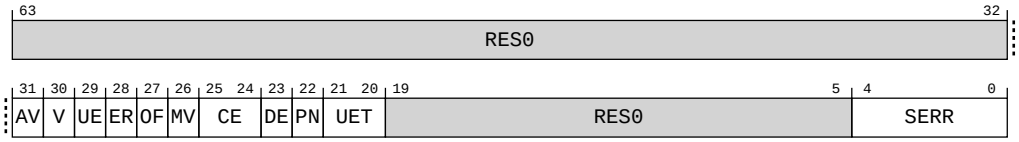


Table A-622: ERXSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	AV	Address Valid. The possible values are: **0b0** ERXADDR_EL1 not valid. **0b1** ERXADDR_EL1 contains an address associated with the highest priority error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[30]	V	Status Register Valid. The possible values are: **0b0** ERXSTATUS_EL1 not valid. **0b1** ERXSTATUS_EL1 valid. At least one error has been recorded. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[29]	UE	Uncorrected Error. The possible values are: **0b0** No errors have been detected, or all detected errors have been either corrected or deferred. **0b1** At least one detected error was not corrected and not deferred. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0

Bits	Name	Description	Reset
[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.	0b0
[27]	OF	Overflow. The possible values are: **0b0** If UE == 1, then no error status for an Uncorrected error has been discarded. If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded. If UE == 0, DE == 0, and CE !== 0b00, then the corrected error counter has not overflowed. **0b1** More than one error has occurred and so details of the other error have been discarded. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. Cold reset only. Unaffected by Warm reset This bit is read/write-one-to-clear.	0b0
[26]	MV	Miscellaneous Registers Valid. The possible values are: **0b0** ERXMISC<m>_EL1 not valid. **0b1** This bit indicates that the ERXMISC<m>_EL1 registers contain additional information for an error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset **Note:** If the ERXMISC<m>_EL1 registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.	0b0

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 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an **UNKNOWN** value. Cold reset only. Unaffected by Warm reset	0b00
[23]	DE	Deferred Error. The possible values are: **0b0** No errors were deferred. **0b1** At least one error was not corrected and deferred. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if ERXSTATUS_EL1.V == 0b0. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[22]	PN	Poison. The value is: **0b0** This core cannot distinguish a poisoned value from a corrupted value. When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads **UNKNOWN** if any of the following are true: • ERXSTATUS_EL1.V == 0b0. • ERXSTATUS_EL1.{DE,UE} == {0,0}. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0

Bits	Name	Description	Reset
[21:20]	UET	Uncorrected Error Type. The value is: **0b00** Uncorrected error, Uncontainable error (UC). Cold reset only. Unaffected by Warm reset	0b00
[19:5]	**RES0**	Reserved	**RES0**
[4:0]	SERR	Primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. The possible values are: **0b00000** No error **0b00010** ECC error from internal data buffer. **0b00110** ECC error on cache data RAM. **0b00111** ECC error on cache tag or dirty RAM. **0b01000** Parity error on TLB data RAM. **0b10010** Error response for a cache copyback. **0b10101** Deferred error from 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	0b00000

Access

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXSTATUS_EL1 is RAZ/WI.
- Direct reads and writes of ERXSTATUS_EL1 are NOPs.
- Direct reads and writes of ERXSTATUS_EL1 are UNDEFINED.

ext-ERR<n>STATUS describes additional constraints that also apply when ext-ERR<n>STATUS is accessed through ERXSTATUS_EL1.

MRS <Xt>, ERXSTATUS_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010

MSR ERXSTATUS_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b010

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXSTATUS_EL1 is RAZ/WI.
- Direct reads and writes of ERXSTATUS_EL1 are NOPs.
- Direct reads and writes of ERXSTATUS_EL1 are UNDEFINED.

ext-ERR<n>STATUS describes additional constraints that also apply when ext-ERR<n>STATUS is accessed through ERXSTATUS_EL1.
MRS <Xt>, ERXSTATUS_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXSTATUS_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXSTATUS_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXSTATUS_EL1;
```

MSR ERXSTATUS_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        ERXSTATUS_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXSTATUS_EL1 = X[t, 64];
```

A.13.6 ERXADDR_EL1, Selected Error Record Address Register

Accesses ext-ERR<n>ADDR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-242: AArch64_erxaddr_el1 bit assignments

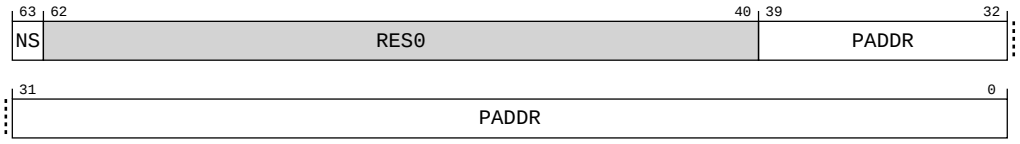


Table A-625: ERXADDR_EL1 bit descriptions

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.	x
[62:40]	**RES0**	Reserved	**RES0**
[39:0]	PADDR	Physical Address. Address of the recorded location. Unaffected by Cold or Warm reset.	40{x}

Access

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXADDR_EL1 is RAZ/WI.
- Direct reads and writes of ERXADDR_EL1 are NOPs.
- Direct reads and writes of ERXADDR_EL1 are UNDEFINED.

ext-ERR<n>ADDR describes additional constraints that also apply when ext-ERR<n>ADDR is accessed through ERXADDR_EL1.

MRS <Xt>, ERXADDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

MSR ERXADDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b011

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXADDR_EL1 is RAZ/WI.
- Direct reads and writes of ERXADDR_EL1 are NOPs.
- Direct reads and writes of ERXADDR_EL1 are UNDEFINED.

ext-ERR<n>ADDR describes additional constraints that also apply when ext-ERR<n>ADDR is accessed through ERXADDR_EL1.
MRS <Xt>, ERXADDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXADDR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXADDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXADDR_EL1;
```

MSR ERXADDR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXADDR_EL1 = X[t, 64];
```

A.13.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register

Accesses ext-ERR<n>PFGF for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x10x xxxx xxxx xxxx xxx0 0000 0110 0010

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-243: AArch64_erxpfgf_el1 bit assignments

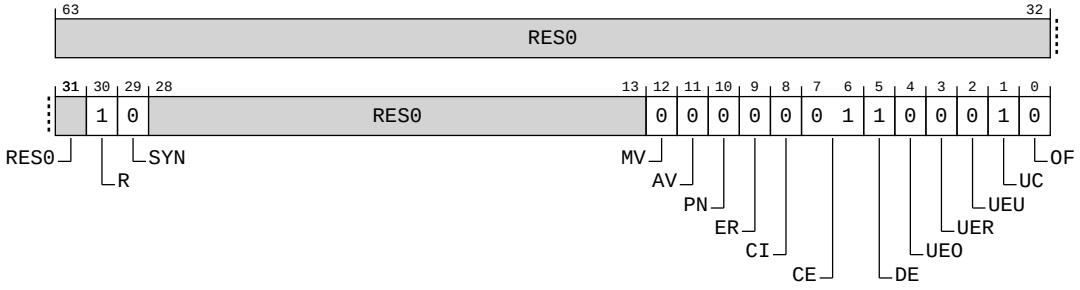


Table A-628: ERXPFGF_EL1 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	R	Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. The value is: **0b1** Feature controllable.	0b1
[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.	0b0
[28:13]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[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 ERXPFGF_EL1.MV == 0b1, software can write specific values into the ERXMISC<m>_EL1 registers when setting up a fault injection event. The values that can be written to these registers are **IMPLEMENTATION DEFINED**.	0b0
[11]	AV	Address syndrome. Address syndrome injection. The value is: **0b0** When an injected error is recorded, the node either sets ERXADDR_EL1 and ERXSTATUS_EL1.AV for the access, or leaves these unchanged.	0b0
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.PN status flag. The value is: **0b0** When an injected error is recorded, the node sets ERXSTATUS_EL1.PN to 0.	0b0
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.ER status flag. The value is: **0b0** When an injected error is recorded, the node sets ERXSTATUS_EL1.ER according to the architecture-defined rules for setting the ER bit.	0b0
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.CI status flag. The value is: **0b0** The node does not support this type of flag This behavior replaces the architecture-defined rules for setting the CI bit.	0b0
[7:6]	CE	Corrected Error generation. The value is: **0b01** The fault generation feature of the node allows generation of a non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10. All other values are reserved.	0b01
[5]	DE	Deferred Error generation. The value is: **0b1** The fault generation feature of the node allows generation of this type of error.	0b1
[4]	UEO	Latent or Restartable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	0b0

Bits	Name	Description	Reset
[3]	UER	Signaled or Recoverable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	0b0
[2]	UEU	Unrecoverable Error generation. The value is: **0b0** The fault generation feature of the node cannot generate this type of error.	0b0
[1]	UC	Uncontainable Error generation. The value is: **0b1** The fault generation feature of the node allows generation of this type of error.	0b1
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.OF status flag. The value is: **0b0** When an injected error is recorded, the node sets ERXSTATUS_EL1.OF according to the architecture-defined rules for setting the OF bit.	0b0

Access

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are **UNDEFINED**.


Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGF is not present, meaning reads of ERXPFGF_EL1 are **RES0**.

ext-ERR<n>PFGF describes additional constraints that also apply when ext-ERR<n>PFGF is accessed through ERXPFGF_EL1.

MRS <Xt>, ERXPFGF_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b100

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are UNDEFINED.


Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGF is not present, meaning reads of ERXPFGF_EL1 are RES0.

ext-ERR<n>PFGF describes additional constraints that also apply when ext-ERR<n>PFGF is accessed through ERXPFGF_EL1.
MRS <Xt>, ERXPFGF_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGF_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGF_EL1;
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = ERXPFGF_EL1;
```

A.13.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register

Accesses ext-ERR<n>PFGCTL for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 00xx xxxx xxxx xxxx xxxx x0x0 000x xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-244: AArch64_erxpfgctl_el1 bit assignments

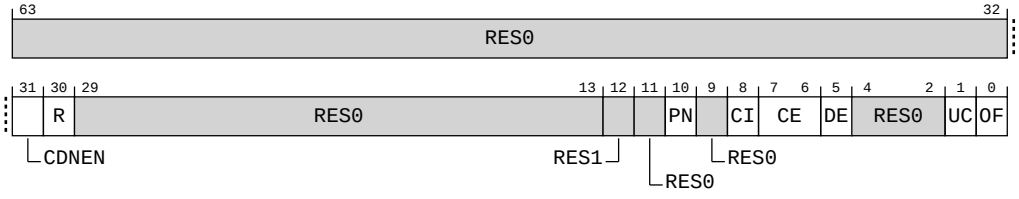


Table A-630: ERXPFGCTL_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	CDNEN	Countdown Enable. Controls transfers from the value that is held in the ERXPFGCDN_EL1 into the Error Generation Counter and enables this counter. **0b0** The Error Generation Counter is disabled. **0b1** The Error Generation Counter is enabled. On a write of 0b1 to this bit, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	0b0
[30]	R	Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. **0b0** On reaching 0, the Error Generation Counter will stop. **0b1** On reaching 0, the Error Generation Counter is set to ERXPFGCDN_EL1.CDN. Cold reset only. Unaffected by Warm reset	0b0
[29:13]	**RES0**	Reserved	**RES0**
[12]	**RES1**	Reserved	**RES1**
[11]	**RES0**	Reserved	**RES0**
[10]	PN	Poison flag. The value that is written to AArch64-ERXSTATUS.PN when an injected error is recorded. **0b00** AArch64-ERXSTATUS.PN is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.PN is set to 1 when an injected error is recorded.	0b0
[9]	**RES0**	Reserved	**RES0**
[8]	CI	Critical Error flag. The value that is written to AArch64-ERXSTATUS.CI when an injected error is recorded. **0b00** AArch64-ERXSTATUS.CI is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.CI is set to 1 when an injected error is recorded.	0b0
[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are: **0b00** No error of this type will be generated. **0b01** A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10, might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b00

Bits	Name	Description	Reset
[5]	DE	Deferred Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[4:2]	**RES0**	Reserved	**RES0**
[1]	UC	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[0]	OF	Overflow flag. The possible values are: **0b0** ERXSTATUS_EL1.OF is set to 0 when an injected error is recorded. **0b1** ERXSTATUS_EL1.OF is set to 1 when an injected error is recorded. Cold reset only. Unaffected by Warm reset	0b0

Access

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are **UNDEFINED**.


Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCTL is not present, meaning reads and writes of ERXPFGCTL_EL1 are **RES0**.

ext-ERR<n>PFGCTL describes additional constraints that also apply when ext-ERR<n>PFGCTL is accessed through ERXPFGCTL_EL1.

MRS <Xt>, ERXPFGCTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

MSR ERXPFGCTL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b101

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCTL_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCTL_EL1 are NOPs.
- Direct reads and writes of ERXPFGCTL_EL1 are UNDEFINED.


Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCTL is not present, meaning reads and writes of ERXPFGCTL_EL1 are RES0.

ext-ERR<n>PFGCTL describes additional constraints that also apply when ext-ERR<n>PFGCTL is accessed through ERXPFGCTL_EL1.
MRS <Xt>, ERXPFGCTL_EL1

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

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCTL_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCTL_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFGCTL_EL1;
```

MSR ERXPFGCTL_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCTL_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXPFGCTL_EL1 = X[t, 64];
```

A.13.9 ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown register

Accesses ext-ERR<n>PFGCDN for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-245: AArch64_erxpfgcdn_el1 bit assignments

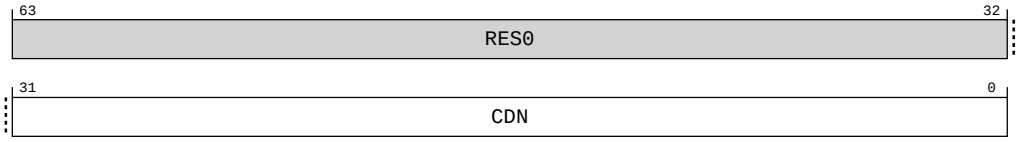


Table A-633: ERXPFGCDN_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	CDN	Countdown value. This field is copied to Error Generation Counter when either: • Software writes ERXPFGCTL_EL1.CDNEN with 1. • The Error Generation Counter decrements to zero and ERXPFGCTL_EL1.R == 0b1. Unaffected by Cold or Warm reset. **Note:** The current Error Generation Counter value is not visible to software.	32{x}

Access

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are **UNDEFINED**.


Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as `0b00`. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCDN is not present, meaning reads and writes of ERXPFGCDN_EL1 are **RES0**.

ext-ERR<n>PFGCDN describes additional constraints that also apply when ext-ERR<n>PFGCDN is accessed through ERXPFGCDN_EL1.

MRS <Xt>, ERXPFGCDN_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

MSR ERXPFGCDN_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b110

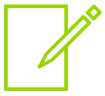
Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGCDN_EL1 is RAZ/WI.
- Direct reads and writes of ERXPFGCDN_EL1 are NOPs.
- Direct reads and writes of ERXPFGCDN_EL1 are UNDEFINED.


Note

A node does not implement the Common Fault Injection Model Extension if ERR<q>FR.INJ reads as 0b00. <q> is the index of the first error record owned by the same node as error record <n>, where <n> is the value in AArch64-ERRSELR_EL1.SEL. If the node owns a single record, then q = n.

If AArch64-ERRSELR_EL1.SEL is not the index of the first error record owned by a node, then ext-ERR<n>PFGCDN is not present, meaning reads and writes of ERXPFGCDN_EL1 are RES0.

ext-ERR<n>PFGCDN describes additional constraints that also apply when ext-ERR<n>PFGCDN is accessed through ERXPFGCDN_EL1.
MRS <Xt>, ERXPFGCDN_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCDN_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXPFGCDN_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXPFGCDN_EL1;
```

MSR ERXPFGCDN_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXPFGCDN_EL1 = X[t, 64];
```

A.13.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0

Accesses ext-ERR<n>MISC0 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-246: AArch64_erxmisc0_el1 bit assignments

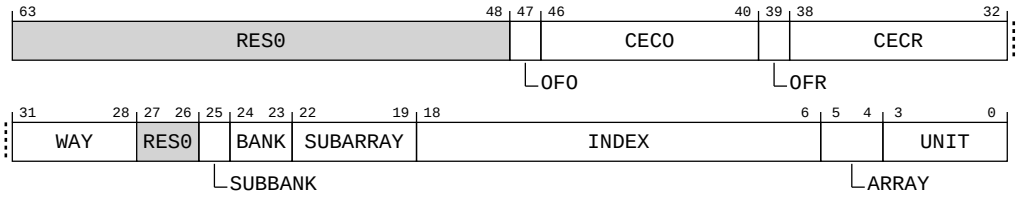


Table A-636: ERXMISC0_EL1 bit descriptions

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

Bits	Name	Description	Reset
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.CECO is incremented and wraps through zero. **0b0** Other counter has not overflowed. **0b1** Other counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR. Unaffected by Cold or Warm reset.	7{x}
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.CECR is incremented and wraps through zero. **0b0** Repeat counter has not overflowed. **0b1** Repeat counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[38:32]	CECR	Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome. This field resets to an IMPLEMENTATION DEFINED which might be **UNKNOWN** on a Cold reset. If the reset value is **UNKNOWN**, then the value of this field remains **UNKNOWN** until software initializes it. Unaffected by Cold or Warm reset.	7{x}
[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.	xxxx
[27:26]	**RES0**	Reserved	**RES0**

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.	x
[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.	xx
[22:19]	SUBARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 data doubleword detected the error. Upper 1 bit is unused. [L1 Data Cache] • Indicates for L1 Data RAM which word had the error detected. For L1 Tag RAMs which bank had the error (0b0000: bank0 , 0b0001: bank1) Unaffected by Cold or Warm reset.	xxxx
[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.	13{x}

Bits	Name	Description	Reset
[5:4]	ARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] Indicates which array detected the error. The possible values are: • `0b00` L2 Tag RAM. • `0b01` L2 Data RAM. • `0b11` CHI Error. [L1 Data Cache] Indicates which array detected the error. The possible values are: • 00 LS Tag RAM 0. • 01 LS Tag RAM 1. • 10 LS Data RAM. • 11 LS Tag RAM 2. [L1 Instruction Cache] Indicates which array that detected the error, Data Array has higher priority. The possible values are: • `0b00` Tag. • `0b01` Data. • `0b10` Macro-OP cache. Unaffected by Cold or Warm reset.	`xx`
[3:0]	UNIT	Indicates the unit which detected the error. The possible values are: **`0b0001`** L1 Instruction Cache. **`0b0010`** L2 TLB. **`0b0100`** L1 Data Cache. **`0b1000`** L2 Cache.	`xxxx`

Access

If AArch64-ERRIDR_EL1.NUM is `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC0_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC0_EL1 are NOPs.
- Direct reads and writes of ERXMISC0_EL1 are UNDEFINED.

ext-ERR<n>MISC0 describes additional constraints that also apply when ext-ERR<n>MISC0 is accessed through ERXMISC0_EL1.

MRS <Xt>, ERXMISC0_EL1

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

MSR ERXMISC0_EL1, <Xt>

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

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC0_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC0_EL1 are NOPs.
- Direct reads and writes of ERXMISC0_EL1 are UNDEFINED.

ext-ERR<n>MISC0 describes additional constraints that also apply when ext-ERR<n>MISC0 is accessed through ERXMISC0_EL1.
MRS <Xt>, ERXMISC0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC0_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC0_EL1;
```

MSR ERXMISC0_EL1, <Xt>

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

```
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC0_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC0_EL1 = X[t, 64];
```

A.13.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1

Accesses ext-ERR<n>MISC1 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-247: AArch64_erxmisc1_el1 bit assignments

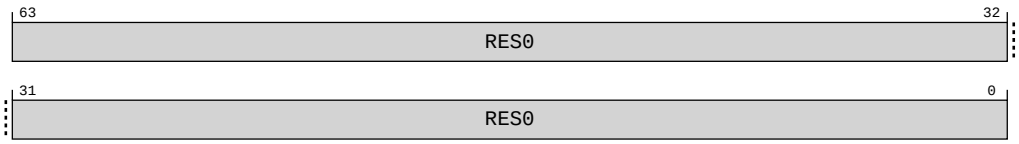


Table A-639: ERXMISC1_EL1 bit descriptions

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

Access

If AArch64-ERRIDR_EL1.NUM is `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC1_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC1_EL1 are NOPs.
- Direct reads and writes of ERXMISC1_EL1 are UNDEFINED.

ext-ERR<n>MISC1 describes additional constraints that also apply when ext-ERR<n>MISC1 is accessed through ERXMISC1_EL1.

MRS <Xt>, ERXMISC1_EL1

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

MSR ERXMISC1_EL1, <Xt>

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

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC1_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC1_EL1 are NOPs.
- Direct reads and writes of ERXMISC1_EL1 are UNDEFINED.

ext-ERR<n>MISC1 describes additional constraints that also apply when ext-ERR<n>MISC1 is accessed through ERXMISC1_EL1.
MRS <Xt>, ERXMISC1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC1_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC1_EL1;
```

MSR ERXMISC1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC1_EL1 = X[t, 64];
```

A.13.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2

Accesses ext-ERR<n>MISC2 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-248: AArch64_erxmisc2_el1 bit assignments

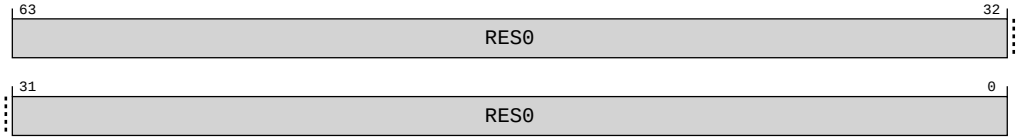


Table A-642: ERXMISC2_EL1 bit descriptions

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

Access

If AArch64-ERRIDR_EL1.NUM is `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC2_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC2_EL1 are NOPs.
- Direct reads and writes of ERXMISC2_EL1 are UNDEFINED.

ext-ERR<n>MISC2 describes additional constraints that also apply when ext-ERR<n>MISC2 is accessed through ERXMISC2_EL1.

MRS <Xt>, ERXMISC2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b010

MSR ERXMISC2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b010

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC2_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC2_EL1 are NOPs.
- Direct reads and writes of ERXMISC2_EL1 are UNDEFINED.

ext-ERR<n>MISC2 describes additional constraints that also apply when ext-ERR<n>MISC2 is accessed through ERXMISC2_EL1.
MRS <Xt>, ERXMISC2_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC2_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC2_EL1;
```

MSR ERXMISC2_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        ERXMISC2_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC2_EL1 = X[t, 64];
```

A.13.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3

Accesses ext-ERR<n>MISC3 for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-249: AArch64_erxmisc3_el1 bit assignments

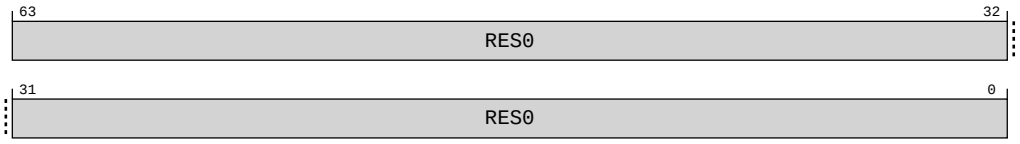


Table A-645: ERXMISC3_EL1 bit descriptions

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

Access

If AArch64-ERRIDR_EL1.NUM is `0x0000` or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An **UNKNOWN** error record is selected.
- ERXMISC3_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC3_EL1 are NOPs.
- Direct reads and writes of ERXMISC3_EL1 are UNDEFINED.

ext-ERR<n>MISC3 describes additional constraints that also apply when ext-ERR<n>MISC3 is accessed through ERXMISC3_EL1.

MRS <Xt>, ERXMISC3_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b011

MSR ERXMISC3_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0101	0b011

Accessibility

If AArch64-ERRIDR_EL1.NUM is 0x0000 or AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then one of the following occurs:

- An UNKNOWN error record is selected.
- ERXMISC3_EL1 is RAZ/WI.
- Direct reads and writes of ERXMISC3_EL1 are NOPs.
- Direct reads and writes of ERXMISC3_EL1 are UNDEFINED.

ext-ERR<n>MISC3 describes additional constraints that also apply when ext-ERR<n>MISC3 is accessed through ERXMISC3_EL1.
MRS <Xt>, ERXMISC3_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC3_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ERXMISC3_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ERXMISC3_EL1;
```

MSR ERXMISC3_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ERXMISC3_EL1 = X[t, 64];
```

A.14 AArch64 Statistical Profiling Extension registers summary

The summary table provides an overview of all Statistical Profiling Extension registers in the core. For more information about a register, click the register name in the table.

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

Table A-648: Statistical Profiling Extension registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSCR_EL1	3	0	C9	C9	0	–	64-bit	Statistical Profiling Control Register (EL1)
PMSICR_EL1	3	0	C9	C9	2	–	64-bit	Sampling Interval Counter Register
PMSIRR_EL1	3	0	C9	C9	3	–	64-bit	Sampling Interval Reload Register
PMSFCR_EL1	3	0	C9	C9	4	–	64-bit	Sampling Filter Control Register
PMSEVFR_EL1	3	0	C9	C9	5	–	64-bit	Sampling Event Filter Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSLATFR_EL1	3	0	C9	C9	6	–	64-bit	Sampling Latency Filter Register
PMSIDR_EL1	3	0	C9	C9	7	–	64-bit	Sampling Profiling ID Register
PMBLIMITR_EL1	3	0	C9	C10	0	–	64-bit	Profiling Buffer Limit Address Register
PMBPTR_EL1	3	0	C9	C10	1	–	64-bit	Profiling Buffer Write Pointer Register
PMBSR_EL1	3	0	C9	C10	3	–	64-bit	Profiling Buffer Status/syndrome Register
PMBIDR_EL1	3	0	C9	C10	7	–	64-bit	Profiling Buffer ID Register
PMSCR_EL2	3	4	C9	C9	0	–	64-bit	Statistical Profiling Control Register (EL2)

A.14.1 PMSEVFR_EL1, Sampling Event Filter Register

Controls sample filtering by events. The overall filter is the logical AND of these filters. For example, if PMSEVFR_EL1.E[3] and PMSEVFR_EL1.E[5] are both set to 1, only samples that have both event 3 (Level 1 unified or data cache refill) and event 5 (TLB walk) set to 1 are recorded.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Statistical Profiling Extension registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 0000 xxxx xxxx xxxx 0xx0 xxxx xxxx x0xx xxx0

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-250: AArch64_pmsevfr_el1 bit assignments

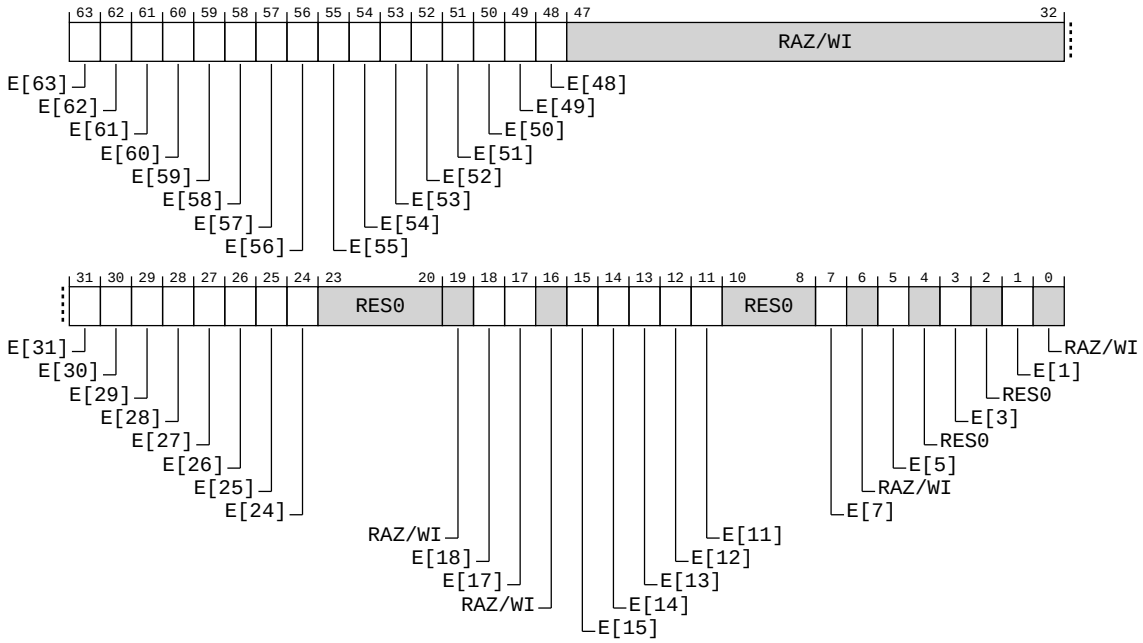


Table A-649: PMSEVFR_EL1 bit descriptions

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	x
[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	x

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	x
[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	x
[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	x
[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	x

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	x
[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	x
[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	x
[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	x

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	x
[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	x
[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	x
[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	x

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	x
[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	x
[47:32]	**RAZ/WI**	Reserved	**RAZ/WI**
[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	x
[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	x

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	x
[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	x
[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	x
[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	x

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	x
[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	x
[23:20]	**RES0**	Reserved	**RES0**
[19]	**RAZ/WI**	Reserved	**RAZ/WI**
[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.	x
[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.	x
[16]	**RAZ/WI**	Reserved	**RAZ/WI**

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	x
[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	x
[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	x
[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	x

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.	x
[10:8]	**RES0**	Reserved	**RES0**
[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.	x
[6]	**RAZ/WI**	Reserved	**RAZ/WI**
[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.	x
[4]	**RES0**	Reserved	**RES0**
[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.	x
[2]	**RES0**	Reserved	**RES0**
[1]	E[1]	Architecturally executed. When the PE supports sampling of speculative instructions: **0b0** Architecturally executed event is ignored. **0b1** Do not record samples that have the Architecturally executed event == 0. This bit is ignored by the PE when AArch64-PMSFCR_EL1.FE == 0. If the PE does not support the sampling of speculative instructions, or always discards the sample record for speculative instructions, this bit reads as an **UNKNOWN** value and the PE ignores its value.	x

Bits	Name	Description	Reset
[0]	RAZ/WI	Reserved	RAZ/WI

Access

MRS <Xt>, PMSEVFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101

MSR PMSEVFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101

Accessibility

MRS <Xt>, PMSEVFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSEVFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSEVFR_EL1;
```

MSR PMSEVFR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSEVFR_EL1 = X[t, 64];
```

A.14.2 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 registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0010 0110 0100 x001 0xxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-251: AArch64_pmsidr_el1 bit assignments

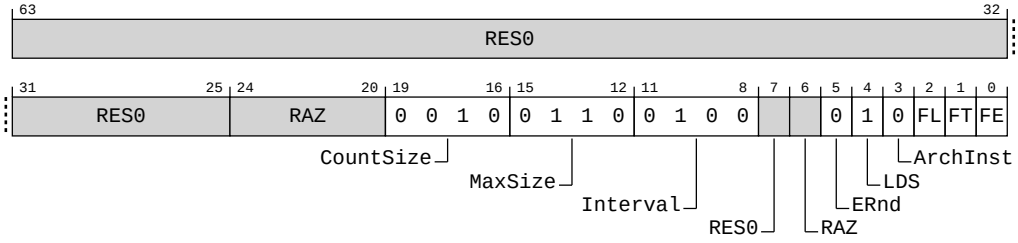


Table A-652: PMSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:25]	**RES0**	Reserved	**RES0**
[24:20]	**RAZ**	Reserved	**RAZ**
[19:16]	CountSize	Defines the size of the counters. **0b0010** 12-bit saturating counters.	0b0010
[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 (AArch64-PMBIDR_EL1.Align), then each record is exactly this size. Defined values are: **0b0110** 64 bytes.	0b0110
[11:8]	Interval	Recommended minimum sampling interval. This provides guidance from the implementer to the smallest minimum sampling interval, N. Defined values are: **0b0100** 1,024.	0b0100
[7]	**RES0**	Reserved	**RES0**
[6]	**RAZ**	Reserved	**RAZ**
[5]	ERnd	Defines how the random number generator is used in determining the interval between samples, when enabled by AArch64-PMSIRR_EL1.RND. Defined values are: **0b0** The random number is added at the start of the interval, and the sample is taken and a new interval started when the combined interval expires.	0b0
[4]	LDS	Data source indicator for sampled load instructions. Defined values are: **0b1** Loaded data source implemented.	0b1
[3]	ArchInst	Architectural instruction profiling. Defined values are: **0b0** Micro-op sampling implemented.	0b0
[2]	FL	Filtering by latency. This bit reads as one.	x
[1]	FT	Filtering by operation type. This bit reads as one.	x
[0]	FE	Filtering by events. This bit reads as one.	x

Access

MRS <Xt>, PMSIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b111

Accessibility

MRS <Xt>, PMSIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && MDCR_EL2.TPMS == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSIDR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSIDR_EL1;
```

A.14.3 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 registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx10 0110


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure A-252: AArch64_pmbidr_el1 bit assignments

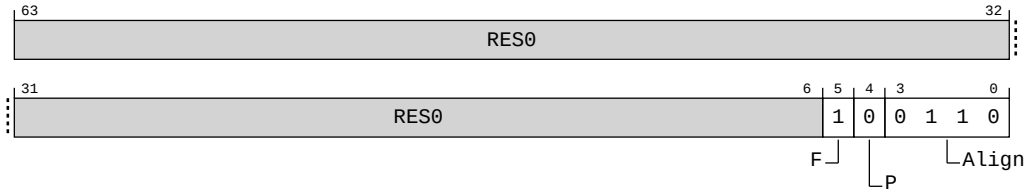


Table A-654: PMBIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:6]	**RES0**	Reserved	**RES0**
[5]	F	Flag updates. Describes how address translations performed by the Statistical Profiling Unit manage the Access flag and dirty state. **0b1** Hardware management of the Access flag and dirty state for accesses made by the Statistical Profiling Unit is controlled in the same way as explicit memory accesses in the Profiling Buffer owning translation regime.	0b1
[4]	P	Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the Profiling Buffer is owned by a higher Exception level or another Security state. Defined values are: **0b0** Programming is allowed.	0b0
[3:0]	Align	Defines the minimum alignment constraint for writes to AArch64-PMBPTR_EL1. Defined values are: **0b0110** 64 bytes.	0b0110

Access

MRS <Xt>, PMBIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1010	0b111

Accessibility

MRS <Xt>, PMBIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMBIDR_EL1;
```

A.15 AArch64 Trace Buffer Extension registers summary

The summary table provides an overview of all Trace Buffer Extension registers in the core. For more information about a register, click the register name in the table.

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

Table A-656: Trace Buffer Extension registers summary

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

Appendix B External registers

This appendix contains the descriptions for the Cortex®-X3 external registers.

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

B.1 External CoreROM registers summary

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

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

Table B-1: CoreROM registers summary

Offset	Name	Reset	Width	Description
0x000	COREROM_ROMENTRY0	—	32-bit	Core ROM table Entry 0
0x004	COREROM_ROMENTRY1	—	32-bit	Core ROM table Entry 1
0x008	COREROM_ROMENTRY2	—	32-bit	Core ROM table Entry 2
0x00C	COREROM_ROMENTRY3	—	32-bit	Core ROM table Entry 3
0xFB8	COREROM_AUTHSTATUS	—	32-bit	Core ROM table Authentication Status Register
0xFBC	COREROM_DEVARCH	—	32-bit	Core ROM table Device Architecture Register
0xFCC	COREROM_DEVTYPE	—	32-bit	Core ROM table Device Type Register
0xFD0	COREROM_PIDR4	—	32-bit	Core ROM table Peripheral Identification Register 4
0xFE0	COREROM_PIDR0	—	32-bit	Core ROM table Peripheral Identification Register 0
0xFE4	COREROM_PIDR1	—	32-bit	Core ROM table Peripheral Identification Register 1
0xFE8	COREROM_PIDR2	—	32-bit	Core ROM table Peripheral Identification Register 2
0xFEC	COREROM_PIDR3	—	32-bit	Core ROM table Peripheral Identification Register 3
0xFF0	COREROM_CIDR0	—	32-bit	Core ROM table Component Identification Register 0
0xFF4	COREROM_CIDR1	—	32-bit	Core ROM table Component Identification Register 1
0xFF8	COREROM_CIDR2	—	32-bit	Core ROM table Component Identification Register 2
0xFFC	COREROM_CIDR3	—	32-bit	Core ROM table Component Identification Register 3

B.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

Access type

RO

Reset value

0000 0000 0000 0001 0000 xxxx xxxx x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-1: ext_corerom_romentry0 bit assignments

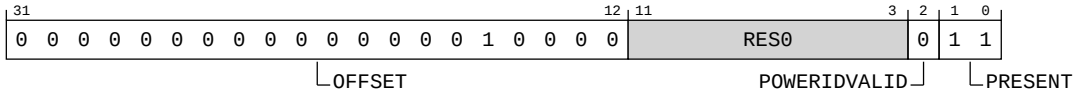


Table B-2: COREROM_ROMENTRY0 bit descriptions

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.	0x00010
[11:3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present.	0b11

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

0000 0000 0000 0010 0000 xxxx xxxx x011

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-2: ext_corerom_romentry1 bit assignments

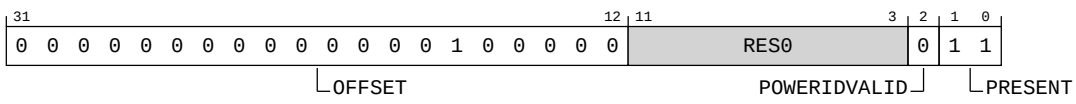


Table B-3: COREROM_ROMENTRY1 bit descriptions

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.	0x00020
[11:3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present.	0b11

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

0000 0000 0000 0011 0000 xxxx xxxx x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-3: ext_corerom_romentry2 bit assignments

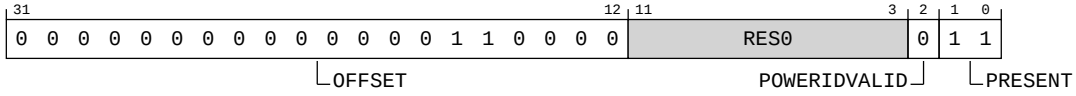


Table B-4: COREROM_ROMENTRY2 bit descriptions

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.	0x00030
[11:3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present.	0b11

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

0000 0000 0000 0100 0000 xxxx xxxx x011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-4: ext_corerom_romentry3 bit assignments

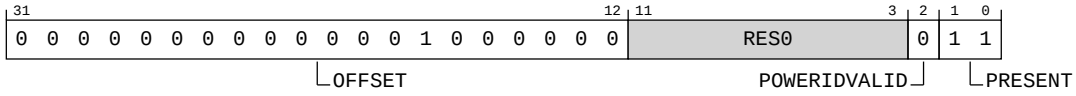


Table B-5: COREROM_ROMENTRY3 bit descriptions

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. When the core is configured without ELA, this field is set to 0x000.	0x00040
[11:3]	**RES0**	Reserved	**RES0**
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present. When the core is configured without ELA, this field is set to 0x0.	0b11

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-5: ext_corerom_authstatus bit assignments

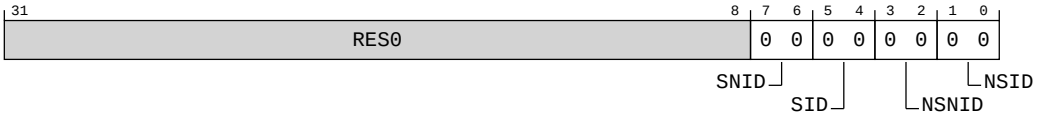


Table B-6: COREROM_AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:6]	SNID	Secure Non-invasive Debug. **0b00** This field is not supported	0b00

Bits	Name	Description	Reset
[5:4]	SID	Secure Invasive Debug. **0b00** This field is not supported	0b00
[3:2]	NSNID	Non-secure Non-invasive Debug. **0b00** Debug level is not supported.	0b00
[1:0]	NSID	Non-secure Invasive Debug. **0b00** Debug level is not supported.	0b00

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

0100 0111 0111 0000 0000 1010 1111 0111

Bit descriptions

Figure B-6: ext_corerom_devarch bit assignments

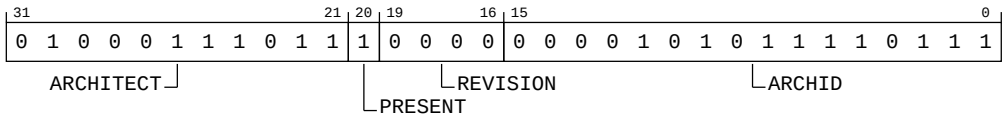


Table B-7: COREROM_DEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	0b01000111011
[20]	PRESENT	Present. **0b1** DEVARCH information present.	0b1
[19:16]	REVISION	Revision. **0b0000** Revision 0.	0b0000
[15:0]	ARCHID	Architecture ID. **0b0000101011110111** ROM Table v0. The debug tool must inspect ext-COREROM_DEVTYPE and ext-COREROM_DEVID to determine further information about the ROM Table.	0x0AF7

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-7: ext_corerom_devtype bit assignments

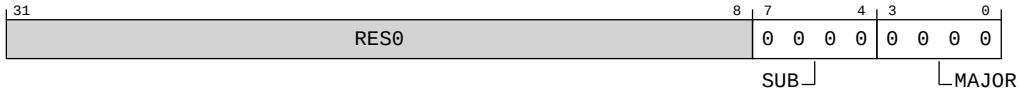


Table B-8: COREROM_DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Sub number **0b0000** Other, undefined.	0b0000
[3:0]	MAJOR	Major number **0b0000** Miscellaneous.	0b0000

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-8: ext_corerom_pidr4 bit assignments

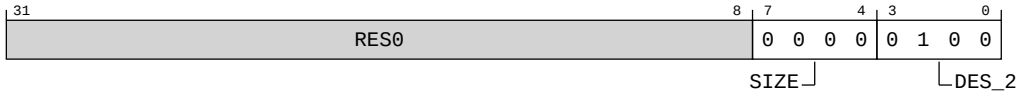


Table B-9: COREROM_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	4KB count. **0b0000** The component uses a single 4KB block.	0b0000
[3:0]	DES_2	JEP106 continuation code. **0b0100** Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B.	0b0100

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0100 1110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-9: ext_corerom_pidr0 bit assignments

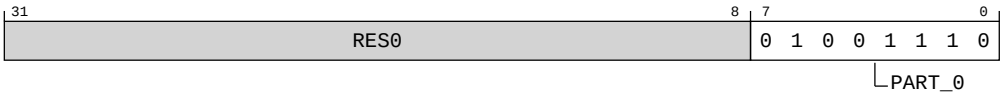


Table B-10: COREROM_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number bits [7:0]. **0b01001110** Cortex®-X3 Core ROM table. Bits [7:0] of part number 0x4E.	0x4E

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-10: ext_corerom_pidr1 bit assignments

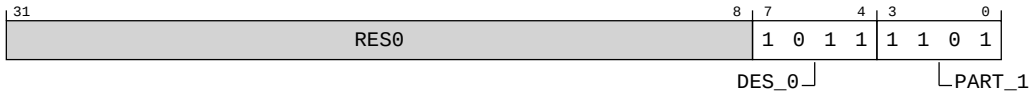


Table B-11: COREROM_PIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	JEP106 identification code bits [3:0]. **0b1011** Arm Limited. Bits [3:0] of JEP106 identification code 0x3B.	0b1011

Bits	Name	Description	Reset
[3:0]	PART_1	Part number bits [11:8]. **0b1101** Cortex®-X3 Core ROM table. Bits [11:8] of part number 0xD.	0b1101

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 1011

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-11: ext_corerom_pidr2 bit assignments

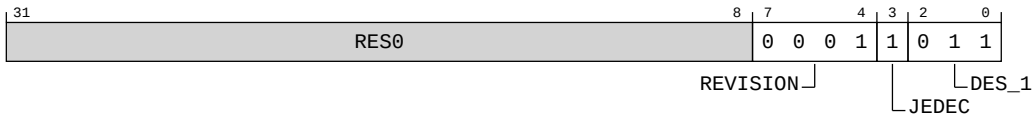


Table B-12: COREROM_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component revision. **0b0001** r1p2	0b0001
[3]	JEDEC	JEDEC assignee. **0b1** JEDEC-assignee values is used.	0b1
[2:0]	DES_1	JEP106 identification code bits [6:4]. **0b011** Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-12: ext_corerom_pidr3 bit assignments

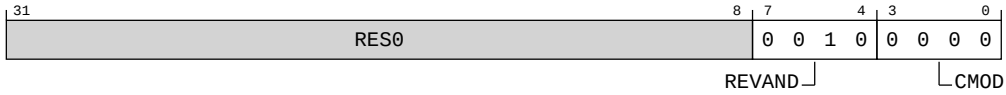


Table B-13: COREROM_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Minor errata fixes. **0b0010**	0b0010
[3:0]	CMOD	Customer Modified. **0b0000** The component is not modified from the original design.	0b0000

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-13: ext_corerom_cidr0 bit assignments

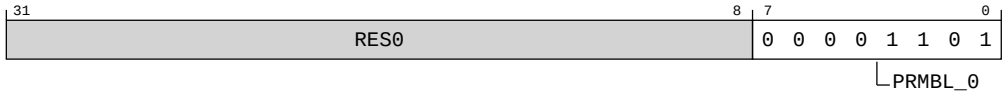


Table B-14: COREROM_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	CoreSight component identification preamble. **0b00001101** CoreSight component identification preamble.	0x0D

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1001 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-14: ext_corerom_cidr1 bit assignments

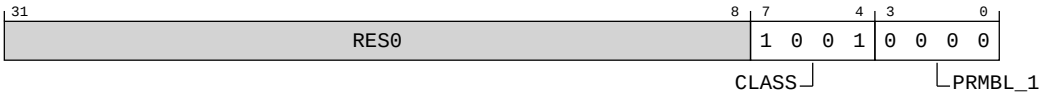


Table B-15: COREROM_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. **0b1001** CoreSight component.	0b1001
[3:0]	PRMBL_1	CoreSight component identification preamble. **0b0000** CoreSight component identification preamble.	0b0000

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-15: ext_corerom_cidr2 bit assignments

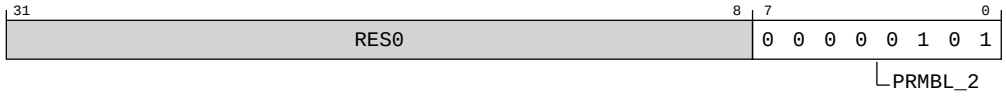


Table B-16: COREROM_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PRMBL_2	CoreSight component identification preamble. **0b00000101** CoreSight component identification preamble.	0x05

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-16: ext_corerom_cidr3 bit assignments

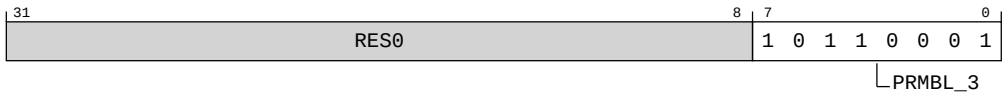


Table B-17: COREROM_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	CoreSight component identification preamble. **0b10110001** CoreSight component identification preamble.	0xB1

Accessibility

This interface is accessible as follows:

RO

B.2 External PPM registers summary

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

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

Table B-18: PPM registers summary

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	—	64-bit	Power Performance Management Register
0x010	CPUPPMCR2	—	64-bit	Power Performance Management Register
0x020	CPUPPMCR3	—	64-bit	Power Performance Management Register
0x080	CPUPPMCR4	—	64-bit	Power Performance Management Register
0x088	CPUPPMCR5	—	64-bit	Power Performance Management Register
0x090	CPUPPMCR6	—	64-bit	Power Performance Management Register

B.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

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-17: ext_cpuppmcr bit assignments

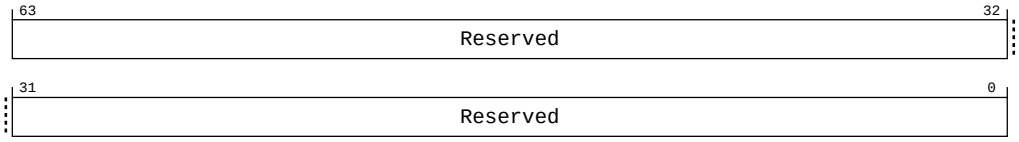


Table B-19: CPUPPMCR bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	64{x}

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-18: ext_cpuppmcr2 bit assignments

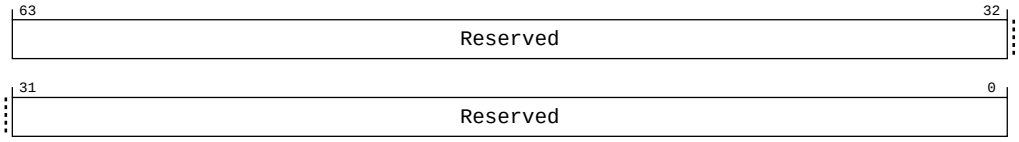


Table B-20: CPUPPMCR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	64{x}

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-19: ext_cpuppmcr3 bit assignments

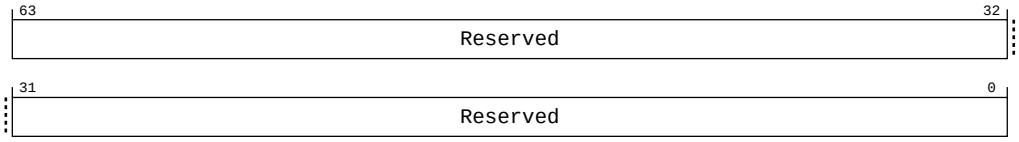


Table B-21: CPUPPMCR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	64{x}

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-20: ext_cpuppmcr4 bit assignments

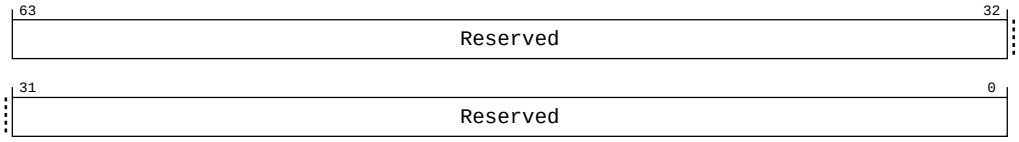


Table B-22: CPUPPMCR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	64{x}

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-21: ext_cpuppmcr5 bit assignments

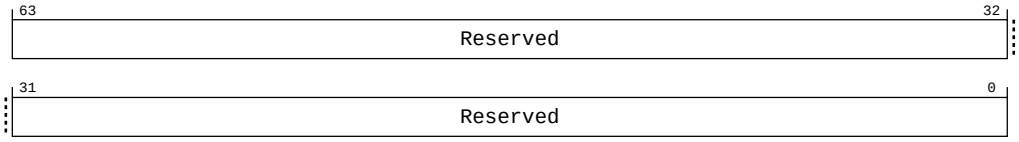


Table B-23: CPUPPMCR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	64{x}

Accessibility

This interface is accessible as follows:

RO

B.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

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-22: ext_cpuppmcr6 bit assignments

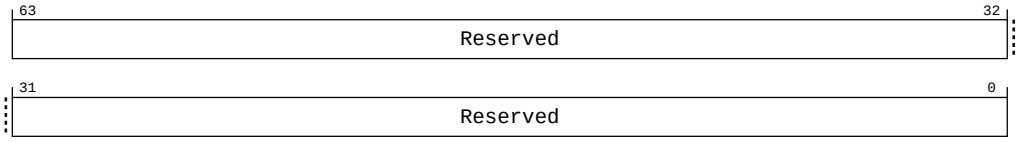


Table B-24: CPUPPMCR6 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	64{x}

Accessibility

This interface is accessible as follows:

RO

B.3 External PMU registers summary

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

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

Table B-25: PMU registers summary

Offset	Name	Reset	Width	Description
0x0	PMEVCNTR0_EL0	—	64-bit	Performance Monitors Event Count Registers
0x8	PMEVCNTR1_EL0	—	64-bit	Performance Monitors Event Count Registers

Offset	Name	Reset	Width	Description
0x10	PMEVCNTR2_EL0	—	64-bit	Performance Monitors Event Count Registers
0x18	PMEVCNTR3_EL0	—	64-bit	Performance Monitors Event Count Registers
0x20	PMEVCNTR4_EL0	—	64-bit	Performance Monitors Event Count Registers
0x28	PMEVCNTR5_EL0	—	64-bit	Performance Monitors Event Count Registers
0x40	PMEVCNTR8_EL0	—	64-bit	Performance Monitors Event Count Registers
0x60	PMEVCNTR12_EL0	—	64-bit	Performance Monitors Event Count Registers
0x80	PMEVCNTR16_EL0	—	64-bit	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_EL0 [31:0]	—	32-bit	Performance Monitors Cycle Counter
0x0FC	PMCCNTR_EL0 [63:32]	—	32-bit	Performance Monitors Cycle Counter
0x200	PMPCSR [31:0]	—	32-bit	Program Counter Sample Register
0x204	PMPCSR [63:32]	—	32-bit	Program Counter Sample Register
0x220	PMPCSR [31:0]	—	32-bit	Program Counter Sample Register
0x224	PMPCSR [63:32]	—	32-bit	Program Counter Sample Register
0x208	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x228	PMCID1SR	—	32-bit	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	—	32-bit	VMID Sample Register
0x22C	PMCID2SR	—	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0 [31:0]	—	32-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0	—	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	—	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	—	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	—	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	—	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	—	64-bit	PMU Cycle Counter Snapshot Register
0x620 + (8 * n)	PMEVCNTSR_n_	—	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMSSCR	—	32-bit	PMU Snapshot Capture Register
0xC00	PMCNTENSET_EL0	—	32-bit	Performance Monitors Count Enable Set register
0xC20	PMCNTENCLR_EL0	—	32-bit	Performance Monitors Count Enable Clear register
0xC40	PMINTENSET_EL1	—	32-bit	Performance Monitors Interrupt Enable Set register
0xC60	PMINTENCLR_EL1	—	32-bit	Performance Monitors Interrupt Enable Clear register
0xC80	PMOVSCLR_EL0	—	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCA0	PMSWINC_EL0	—	32-bit	Performance Monitors Software Increment register

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

B.3.1 PMEVCNTR0_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x0

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-23: ext_pmevcntr0_el0 bit assignments

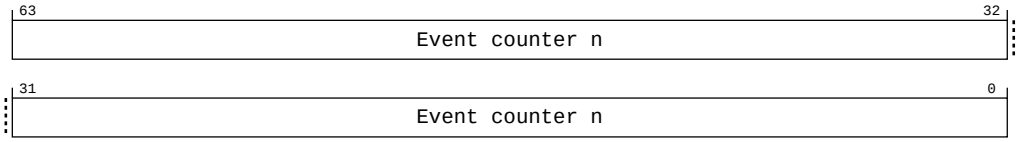


Table B-26: PMEVCNTR0_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	`64{x}`

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x0	PMEVCNTR0_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.2 PMEVCNTR1_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x8

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-24: ext_pmevcntr1_el0 bit assignments

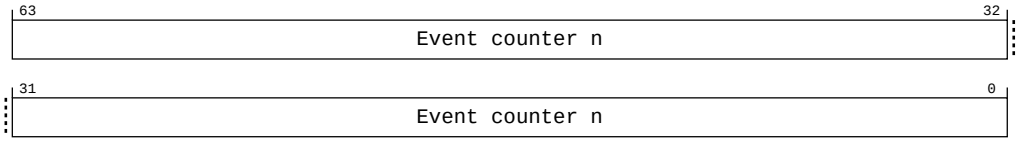


Table B-28: PMEVCNTR1_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	64{x}

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x8	PMEVCNTR1_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.3 PMEVCNTR2_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x10

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-25: ext_pmevcntr2_el0 bit assignments

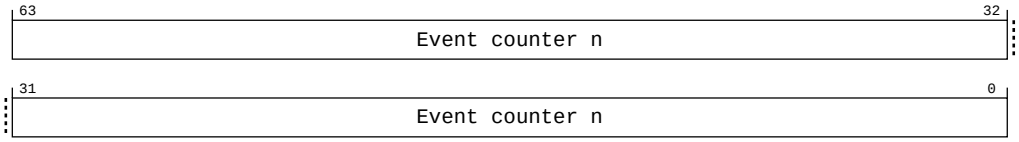


Table B-30: PMEVCNTR2_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	`64{x}`

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.

- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x10	PMEVCNTR2_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.4 PMEVCNTR3_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x18

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-26: ext_pmevcntr3_el0 bit assignments

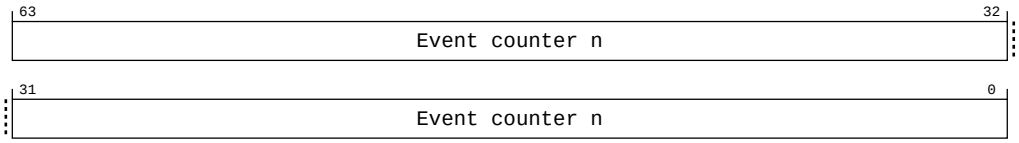


Table B-32: PMEVCNTR3_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	`64{x}`

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x18	PMEVCNTR3_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise
ERROR

B.3.5 PMEVCNTR4_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width
64

Component
PMU

Register offset
0x20

Access type
See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-27: ext_pmevcntr4_el0 bit assignments

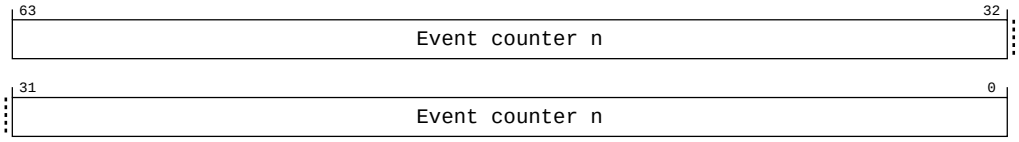


Table B-34: PMEVCNTR4_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	`64{x}`

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x20	PMEVCNTR4_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.6 PMEVCNTR5_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x28

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-28: ext_pmevcntr5_el0 bit assignments

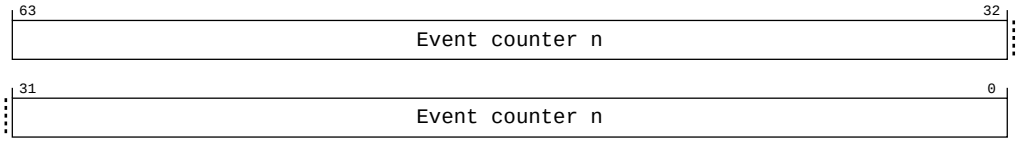


Table B-36: PMEVCNTR5_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	`64{x}`

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x28	PMEVCNTR5_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.7 PMEVCNTR8_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x40

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-29: ext_pmevcntr8_el0 bit assignments

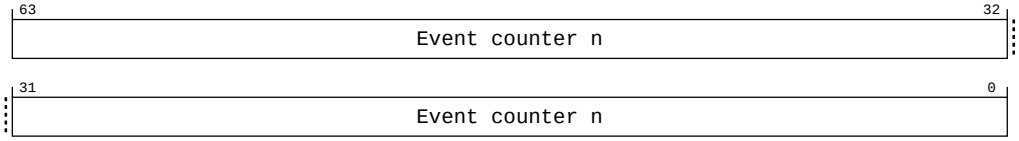


Table B-38: PMEVCNTR8_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	`64{x}`

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004`+8×n have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x40	PMEVCNTR8_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.8 PMEVCNTR12_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x60

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-30: ext_pmevcntr12_el0 bit assignments

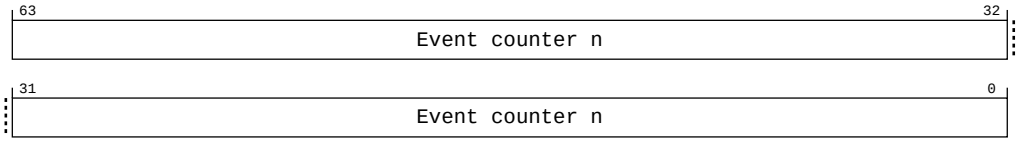


Table B-40: PMEVCNTR12_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	`64{x}`

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x60	PMEVCNTR12_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.9 PMEVCNTR16_EL0, Performance Monitors Event Count Registers

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x80

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-31: ext_pmevcntr16_el0 bit assignments

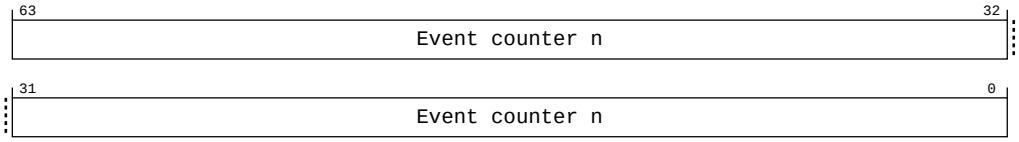


Table B-42: PMEVCNTR16_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30. If the highest implemented Exception level is using AArch32, the optional external interface to the performance monitors is implemented, and the AArch32-PMCR.LP and AArch32-HDCR.HLP bits are RAZ/WI, then locations in the external interface to the performance monitors that map to PMEVCNTR<n>_EL0[63:32] return **UNKNOWN** values on reads. If the implementation does not support AArch64, bits [63:32] of the event counters are not required to be implemented.	64{x}

Access

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to `0x004+8×n` have a **CONSTRAINED UNPREDICTABLE** behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

External accesses to the performance monitors ignore the following controls:

- AArch64-PMUSERENR_EL0.
- If implemented, AArch64-MDCR_EL2.{TPM, TPMCR, HPMN}.
- AArch64-MDCR_EL3.TPM.

This means that all counters are accessible regardless of the current Exception level or privilege of the access.

If FEAT_PMUv3p5 is not implemented, when IsCorePowered(), DoubleLockStatus(), OSLockStatus() or !AllowExternalPMUAccess(), 32-bit accesses to 0x004+8×n have a CONSTRAINED UNPREDICTABLE behavior.

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x80	PMEVCNTR16_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.10 PMCCNTR_EL0, Performance Monitors Cycle Counter

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. For more information, see *Time as measured by the Performance Monitors cycle counter* in the *Arm® Architecture Reference Manual for A-profile architecture*.

ext-PMCCFILTR_EL0 determines the modes and states in which the PMCCNTR_EL0 can increment.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offsets (2)

0x0F8,0x0FC

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-32: ext_pmccntr_el0 bit assignments

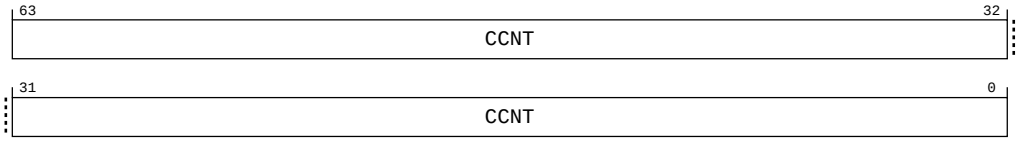


Table B-44: PMCCNTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	CCNT	Cycle count. Depending on the values of ext-PMCR_EL0.{LC,D}, the cycle count increments in one of the following ways: • Every processor clock cycle. • Every 64th processor clock cycle. Writing 1 to ext-PMCR_EL0.C sets this field to 0.	64{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x0F8	PMCCNTR_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

Component	Offset	Instance	Range
PMU	0x0FC	PMCCNTR_EL0	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.11 PMPCSR, Program Counter Sample Register

Holds a sampled instruction address value.

Configurations

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.

Support for 64-bit atomic reads is IMPLEMENTATION DEFINED. If 64-bit atomic reads are implemented, a 64-bit read of PMPCSR has the same side-effect as a 32-bit read of PMCSR[31:0]

followed by a 32-bit read of PMPCSR[63:32], returning the combined value. For example, if the PE is in Debug state then a 64-bit atomic read returns bits[31:0] == 0xFFFFFFFF and bits[63:32] UNKNOWN.

Attributes

Width

64

Component

PMU

Register offsets (4)

0x200,0x204,0x220,0x224

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-33: ext_pmpcsr bit assignments

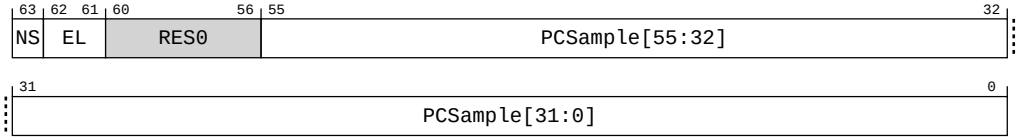


Table B-47: PMPCSR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure state sample. Indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample. **0b0** Sample is from Secure state. **0b1** Sample is from Non-secure state.	x

Bits	Name	Description	Reset
[62:61]	EL	Exception level status sample. Indicates the Exception level that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample. **0b00** Sample is from EL0. **0b01** Sample is from EL1. **0b10** Sample is from EL2. **0b11** Sample is from EL3.	xx
[60:56]	**RES0**	Reserved	**RES0**
[55:32]	PCSample[55:32]	Bits[55:32] of the sampled instruction address value. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.	24{x}
[31:0]	PCSample[31:0]	Bits[31:0] of the sampled instruction address value. PMPCSR[31:0] reads as 0xFFFFFFFF when any of the following are true: • The PE is in Debug state. • PC Sample-based profiling is prohibited. If a branch instruction has retired since the PE left reset state, then the first read of PMPCSR[31:0] is permitted but not required to return 0xFFFFFFFF. PMPCSR[31:0] reads as an **UNKNOWN** value when any of the following are true: • The PE is in reset state. • No branch instruction has retired since the PE left reset state, Debug state, or a state where PC Sample-based Profiling is prohibited. • No branch instruction has retired since the last read of PMPCSR[31:0]. For the cases where a read of PMPCSR[31:0] returns 0xFFFFFFFF or an **UNKNOWN** value, the read has the side-effect of setting PMPCSR[63:32], ext-PMCID1SR, ext-PMCID2SR, and ext-PMVIDSR to **UNKNOWN** values. Otherwise, a read of PMPCSR[31:0] returns bits [31:0] of the sampled instruction address value and has the side-effect of indirectly writing to PMPCSR[63:32], ext-PMCID1SR, ext-PMCID2SR, and ext-PMVIDSR. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}. For a read of PMPCSR[31:0] from the memory-mapped interface, if PMLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the side-effect of the access does not occur and PMPCSR[63:32], ext-PMCID1SR, ext-PMCID2SR, and ext-PMVIDSR are unchanged.	32{x}

Access

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register **UNKNOWN**, see *Permitted behavior that might make the PC Sample-based profiling registers* **UNKNOWN** in the *Arm® Architecture Reference Manual for A-profile architecture*.


Note

A 32-bit access to PMPCSR[63:32] does not update the PC sample registers. Only a 64-bit access to PMPCSR[63:0] or a 32-bit access to PMPCSR[31:0] updates the PC sample registers. This includes the value a subsequent 32-bit read of PMPCSR[63:32] will return.

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.


Note

A 32-bit access to PMPCSR[63:32] does not update the PC sample registers. Only a 64-bit access to PMPCSR[63:0] or a 32-bit access to PMPCSR[31:0] updates the PC sample registers. This includes the value a subsequent 32-bit read of PMPCSR[63:32] will return.

Component	Offset	Instance	Range
PMU	0x200	PMPCSR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
PMU	0x204	PMPCSR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
PMU	0x220	PMPCSR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
PMU	0x224	PMPCSR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

B.3.12 PMCID1SR, CONTEXTIDR_EL1 Sample Register

Contains the sampled value of AArch64-CONTEXTIDR_EL1, captured on reading ext-PMPCSR[31:0].

Configurations

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 offsets (2)

0x208,0x228

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-34: ext_pmcid1sr bit assignments

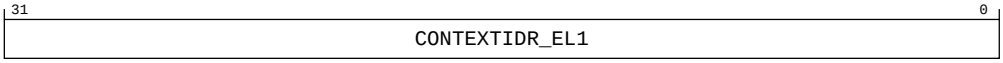


Table B-52: PMCID1SR bit descriptions

Bits	Name	Description	Reset
[31:0]	CONTEXTIDR_EL1	Context ID. The value of CONTEXTIDR that is associated with the most recent ext-PMPCSR sample. When the most recent ext-PMPCSR sample is generated: • If EL1 is using AArch64, then the Context ID is sampled from AArch64-CONTEXTIDR_EL1. Because the value written to PMCID1SR is an indirect read of CONTEXTIDR, it is **CONSTRAINED UNPREDICTABLE** whether PMCID1SR is set to the original or new value if ext-PMPCSR samples: • An instruction that writes to CONTEXTIDR. • The next Context synchronization event. • Any instruction executed between these two instructions.	32{x}

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Component	Offset	Instance	Range
PMU	0x208	PMCID1SR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
PMU	0x228	PMCID1SR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

B.3.13 PMVIDSR, VMID Sample Register

Contains the sampled VMID value that is captured on reading ext-PMPCSR[31:0].

Configurations

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

0x20C

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-35: ext_pmvidsr bit assignments

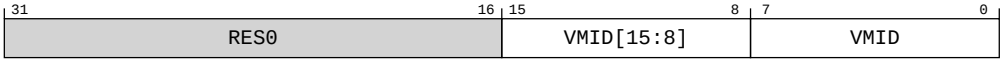


Table B-55: PMVIDSR bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:8]	VMID[15:8]	Extension to VMID[7:0]. For more information, see VMID[7:0].	8{x}

Bits	Name	Description	Reset
[7:0]	VMID	VMID sample. The VMID associated with the most recent ext-PMPCSR sample. When the most recent ext-PMPCSR sample was generated: • This field is set to an **UNKNOWN** value if any of the following apply: ◦ EL2 is disabled in the current Security state. ◦ The PE is executing at EL2. ◦ EL2 is enabled in the current Security state, the PE is executing at EL0, EL2 is using AArch64, HCR_EL2.E2H == 1, and HCR_EL2.TGE == 1. • Otherwise: ◦ If EL2 is using AArch64 and either FEAT_VMID16 is not implemented or AArch64-VTCR_EL2.VS is 1, this field is set to AArch64-VTTBR_EL2.VMID. ◦ If EL2 is using AArch64, FEAT_VMID16 is implemented, and AArch64-VTCR_EL2.VS is 0, PMVIDSR.VMID[7:0] is set to AArch64-VTTBR_EL2.VMID[7:0] and PMVIDSR.VMID[15:8] is RES0. Because the value written to PMVIDSR is an indirect read of the VMID value, it is CONSTRAINED UNPREDICTABLE whether PMVIDSR is set to the original or new value if ext-PMPCSR samples: • An instruction that writes to the VMID value. • The next Context synchronization event. • Any instruction executed between these two instructions.	`8{x}`

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Component	Offset	Instance	Range
PMU	0x20C	PMVIDSR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

B.3.14 PMCID2SR, CONTEXTIDR_EL2 Sample Register

Contains the sampled value of AArch64-CONTEXTIDR_EL2, captured on reading ext-PMPCSR[31:0].

Configurations


Note

If FEAT_PCSRv8p2 is not implemented, 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

0x22C

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-36: ext_pmcid2sr bit assignments

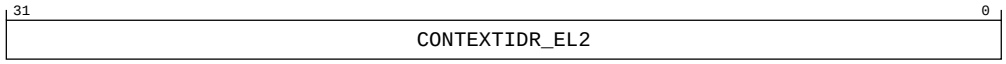


Table B-57: PMCID2SR bit descriptions

Bits	Name	Description	Reset
[31:0]	CONTEXTIDR_EL2	Context ID. The value of AArch64-CONTEXTIDR_EL2 that is associated with the most recent ext-PMPCSR sample. When the most recent ext-PMPCSR sample is generated: • If the PE is not executing at EL3, EL2 is using AArch64, and EL2 is enabled in the current Security state, then this field is set to the Context ID sampled from AArch64-CONTEXTIDR_EL2. • Otherwise, this field is set to an **UNKNOWN** value. Because the value written to PMCID2SR is an indirect read of AArch64-CONTEXTIDR_EL2, it is CONSTRAINED UNPREDICTABLE whether PMCID2SR is set to the original or new value if ext-PMPCSR samples: • An instruction that writes to AArch64-CONTEXTIDR_EL2. • The next Context synchronization event. • Any instruction executed between these two instructions.	32{x}

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Component	Offset	Instance	Range
PMU	0x22C	PMCID2SR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

B.3.15 PMEVTYPER0_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x400

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-37: ext_pmevtyper0_el0 bit assignments

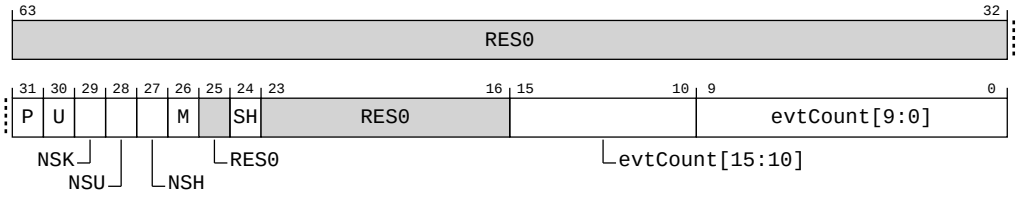


Table B-59: PMEVTYPER0_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x

Bits	Name	Description	Reset
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x400	PMEVTYPER0_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.16 PMEVTYPER1_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x404

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-38: ext_pmevtyper1_el0 bit assignments

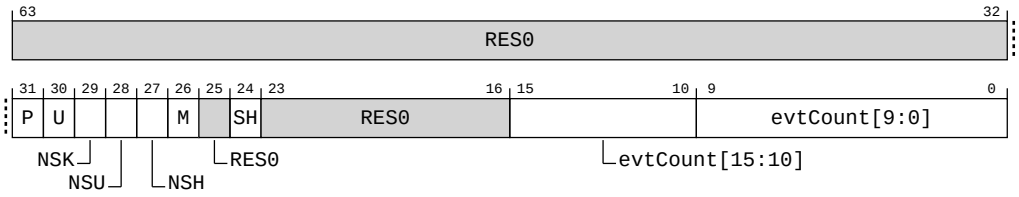


Table B-61: PMEVTYPER1_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}

Bits	Name	Description	Reset
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	`10{x}`

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x404	PMEVTYPER1_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.17 PMEVTYPER2_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x408

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-39: ext_pmevtyper2_el0 bit assignments

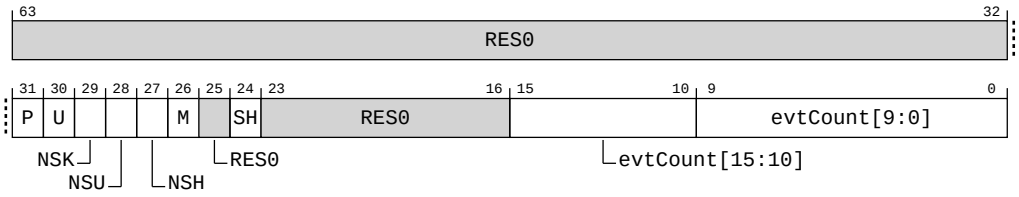


Table B-63: PMEVTYPER2_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x408	PMEVTYPER2_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.18 PMEVTYPER3_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x40C

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-40: ext_pmevtyper3_el0 bit assignments

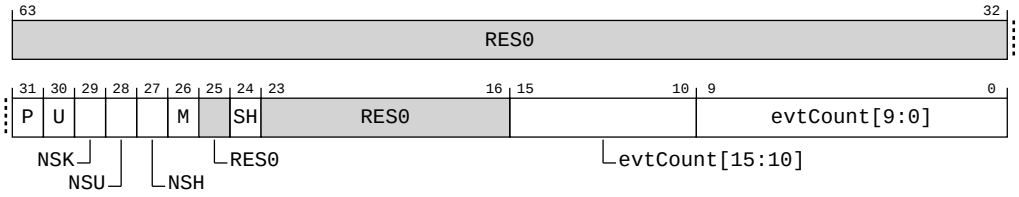


Table B-65: PMEVTYPER3_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x

Bits	Name	Description	Reset
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x40C	PMEVTYPER3_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.19 PMEVTYPER4_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x410

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-41: ext_pmevtyper4_el0 bit assignments

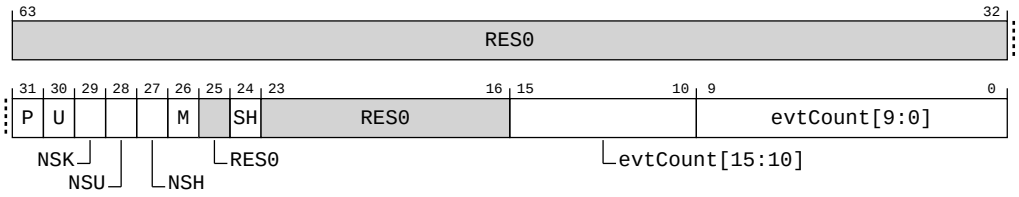


Table B-67: PMEVTYPER4_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}

Bits	Name	Description	Reset
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	`10{x}`

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x410	PMEVTYPER4_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.20 PMEVTYPER5_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x414

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-42: ext_pmevtyper5_el0 bit assignments

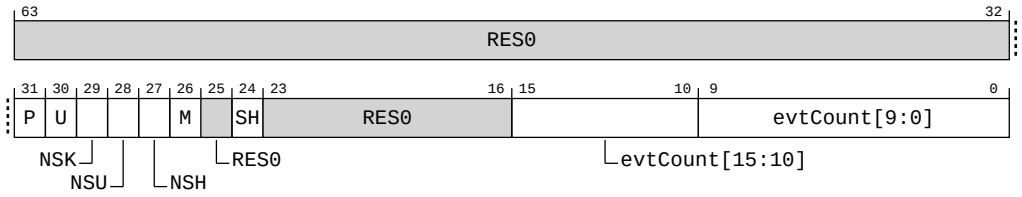


Table B-69: PMEVTYPER5_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x414	PMEVTYPER5_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.21 PMEVTYPER8_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x420

Access type

See bit descriptions

Reset value

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

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-43: ext_pmevtyper8_el0 bit assignments

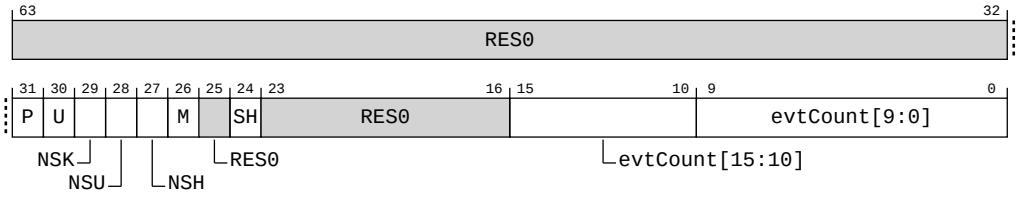


Table B-71: PMEVTYPER8_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x

Bits	Name	Description	Reset
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x420	PMEVTYPER8_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.22 PMEVTYPER12_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x430

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-44: ext_pmevtyper12_el0 bit assignments

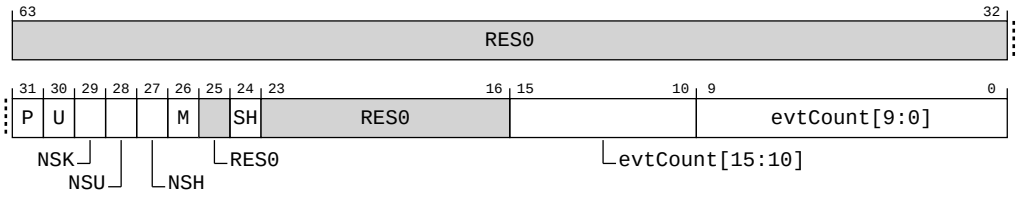


Table B-73: PMEVTYPER12_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}

Bits	Name	Description	Reset
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	`10{x}`

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x430	PMEVTYPER12_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.23 PMEVTYPER16_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess(), accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

Attributes

Width

64

Component

PMU

Register offset

0x440

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-45: ext_pmevtyper16_el0 bit assignments

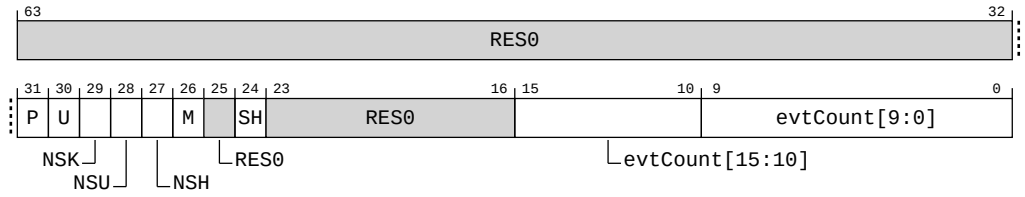


Table B-75: PMEVTYPER16_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMEVTYPER<n>_EL0.NSK bit. **0b0** Count events in EL1. **0b1** Do not count events in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMEVTYPER<n>_EL0.NSU bit. **0b0** Count events in EL0. **0b1** Do not count events in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Non-secure EL1 are counted. Otherwise, events in Non-secure EL1 are not counted.	x
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Non-secure EL0 are counted. Otherwise, events in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMEVTYPER<n>_EL0.SH bit. **0b0** Do not count events in EL2. **0b1** Count events in EL2.	x
[26]	M	EL3 filtering bit. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in EL3 are counted. Otherwise, events in EL3 are not counted. Most applications can ignore this field and set its value to 0b0.	x
[25]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Secure EL2 are counted. Otherwise, events in Secure EL2 are not counted.	x
[23:16]	**RES0**	Reserved	**RES0**
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6{x}
[9:0]	evtCount[9:0]	Event to count. The event number of the event that is counted by event counter ext-PMEVCNTR<n>_EL0. Software must program this field with an event that is supported by the PE being programmed. The ranges of event numbers allocated to each type of event are shown in *Allocation of the PMU event number space* in the *Arm® Architecture Reference Manual for A-profile architecture*. If PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: • For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • If FEAT_PMUv3p1 is implemented, for the range 0x4000 to 0x403F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field. • For other values, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is **UNKNOWN**. **Note:** UNPREDICTABLE means the event must not expose privileged information.	10{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x440	PMEVTYPER16_EL0	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.24 PMCCFILTR_EL0, Performance Monitors Cycle Counter Filter Register

Determines the modes in which the Cycle Counter, ext-PMCCNTR_EL0, increments.

Configurations

On a Warm or Cold reset, RW fields in this register reset to:

- Architecturally UNKNOWN values if the reset is to an Exception level that is using AArch64.

The register is not affected by an External debug reset.

Attributes

Width

32

Component

PMU

Register offset

0x47C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-46: ext_pmccfiltr_el0 bit assignments

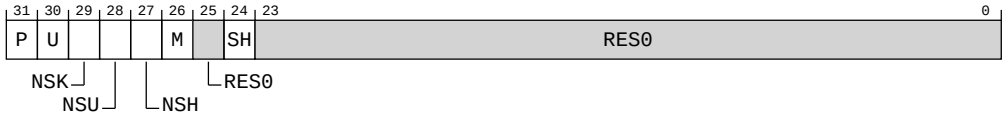


Table B-77: PMCCFILTR_EL0 bit descriptions

Bits	Name	Description	Reset
[31]	P	Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the PMCCFILTR_EL0.NSK bit. **0b0** Count cycles in EL1. **0b1** Do not count cycles in EL1.	x
[30]	U	User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the PMCCFILTR_EL0.NSU bit. **0b0** Count cycles in EL0. **0b1** Do not count cycles in EL0.	x
[29]	NSK	Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Non-secure EL1 are counted. Otherwise, cycles in Non-secure EL1 are not counted.	x
[28]	NSU	Non-secure EL0 (Unprivileged) filtering bit. Controls counting in Non-secure EL0. If the value of this bit is equal to the value of the PMCCFILTR_EL0.U bit, cycles in Non-secure EL0 are counted. Otherwise, cycles in Non-secure EL0 are not counted.	x
[27]	NSH	EL2 (Hypervisor) filtering bit. Controls counting in EL2. If FEAT_SEL2 and EL3 are implemented, counting in Secure EL2 is further controlled by the PMCCFILTR_EL0.SH bit. **0b0** Do not count cycles in EL2. **0b1** Count cycles in EL2.	x
[26]	M	Secure EL3 filtering bit. If the value of this bit is equal to the value of the PMCCFILTR_EL0.P bit, cycles in Secure EL3 are counted. Otherwise, cycles in Secure EL3 are not counted. Most applications can ignore this field and set its value to 0.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. If the value of this bit is not equal to the value of the PMCCFILTR_EL0.NSH bit, cycles in Secure EL2 are counted. Otherwise, cycles in Secure EL2 are not counted.	x
[23:0]	**RES0**	Reserved	**RES0**

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0x47C	PMCCFILTR_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.25 PMPCSSR, Snapshot Program Counter Sample Register

Captured copy of the Program Counter.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x600

Access type

Read

R

Write

RESERVED

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-47: ext_pmpcssr bit assignments

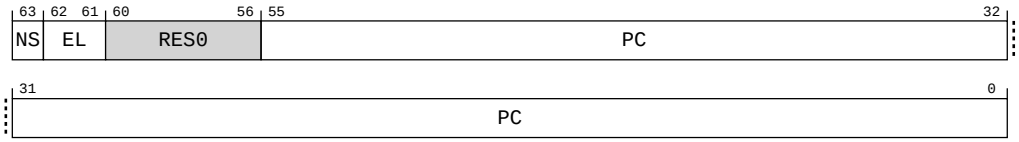


Table B-79: PMPCSSR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure sample. **0b0** The captured instruction was executed in Secure state. **0b1** The captured instruction was executed in Non-secure state.	x
[62:61]	EL	Exception level sample. The Exception level the captured instruction was executed at.	xx
[60:56]	**RES0**	Reserved	**RES0**
[55:0]	PC	Sampled PC. The instruction address for the sampled instruction. The sampled instruction must be an instruction recently executed by the PE. The architecture does not require that all instructions are eligible for sampling. 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.	56{x}

Accessibility

PMPCSSR is set to an UNKNOWN value by a read of the EDPCSRlo or PMPCSR[31:0] register.
This interface is accessible as follows:

RO

B.3.26 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register

Captured copy of the CONTEXTIDR_EL1 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x608

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-48: ext_pmcidssr bit assignments

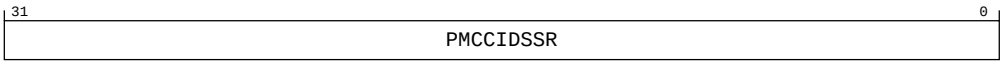


Table B-80: PMCIDSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCIDSSR	PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot.	32{x}

Accessibility

PMCIDSSR is set to an UNKNOWN value by a read of the EDPCSRlo or PMPCSR[31:0] register.
This interface is accessible as follows:

RO

B.3.27 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register

Captured copy of the CONTEXTIDR_EL2 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x60C

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-49: ext_pmcid2ssr bit assignments

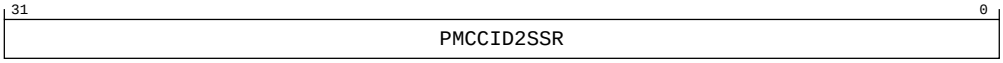


Table B-81: PMCID2SSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCID2SSR	PMCID2SR sample. Sampled CONTEXTIDR_EL2 snapshot.	32{x}

Access

If ARMv8.2 is not implemented, this location is used for PMVIDSSR.

PMCID2SSR is set to an **UNKNOWN** value by a read of the EDPCSRlo or PMPCSR[31:0] register.

Accessibility

If ARMv8.2 is not implemented, this location is used for PMVIDSSR.

PMCID2SSR is set to an UNKNOWN value by a read of the EDPCSRlo or PMPCSR[31:0] register.
This interface is accessible as follows:

RO

B.3.28 PMSSSR, PMU Snapshot Status Register

Holds status information about the captured counters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x610

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx1

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-50: ext_pmsssr bit assignments

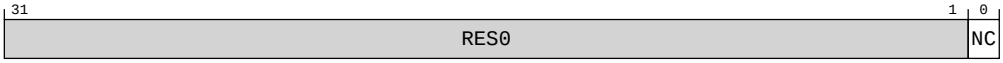


Table B-82: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	NC	No capture. Indicates whether the PMU counters have been captured. **0b0** PMU counters captured. **0b1** PMU counters not captured. The event counters are only not captured by the PE in the event of a security violation. The external Monitor is responsible for keeping track of whether it managed to capture the snapshot registers from the PE. PMSSR.NC does not reflect the status of the captured Program Counter Sample registers. PMSSR.NC is reset to 1 by PE Warm reset, but is overwritten at the first capture. Tools need to be aware that capturing over reset or power-down might lose data, as they are reliant on software saving and restoring the PMU state (including PMSSCR). There is no sampled sticky reset bit.	0b1

Accessibility

This interface is accessible as follows:

RO

B.3.29 PMCCNTSR, PMU Cycle Counter Snapshot Register

Captured copy of PMCCNTR_EL0. Once captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR_EL0 and PMCR_EL0.C.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x618

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-51: ext_pmccntsr bit assignments

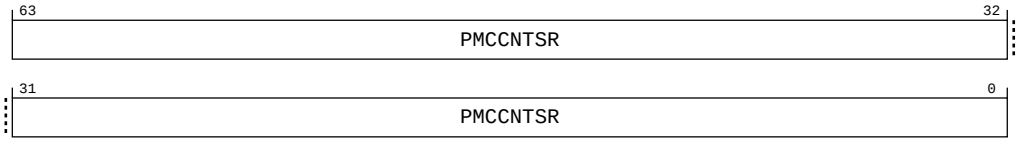


Table B-83: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_EL0 sample. Sampled cycle count.	64{x}

Accessibility

This interface is accessible as follows:

RO

B.3.30 PMEVCNTSR<n>, PMU Event Counter Snapshot Register, n = 0 - 30

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 + (8 * n)

Access type

RO

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-52: ext_pmevcntsr_n_ bit assignments

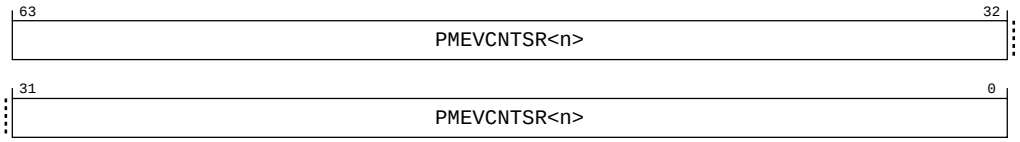


Table B-84: PMEVCNTSR<n> bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x620 + (8 * n)	PMEVCNTSR<n>	None

This interface is accessible as follows:

RO

B.3.31 PMSSCR, PMU Snapshot Capture Register

Provides a mechanism for software to initiate a sample.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x6F0

Access type

RESERVEDW

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-53: ext_pmsscr bit assignments

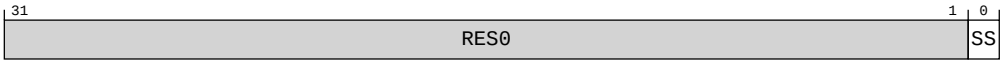


Table B-86: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	SS	Capture now. **0b0** Ignored. **0b1** Initiate a capture immediately.	x

Accessibility

This interface is accessible as follows:

WO

B.3.32 PMCNTENSET_EL0, Performance Monitors Count Enable Set register

Enables the Cycle Count Register, ext-PMCCNTR_EL0, and any implemented event counters ext-PMEVCNTR<n>_EL0. Reading this register shows which counters are enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

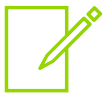
0xC00

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-54: ext_pmcntenset_el0 bit assignments

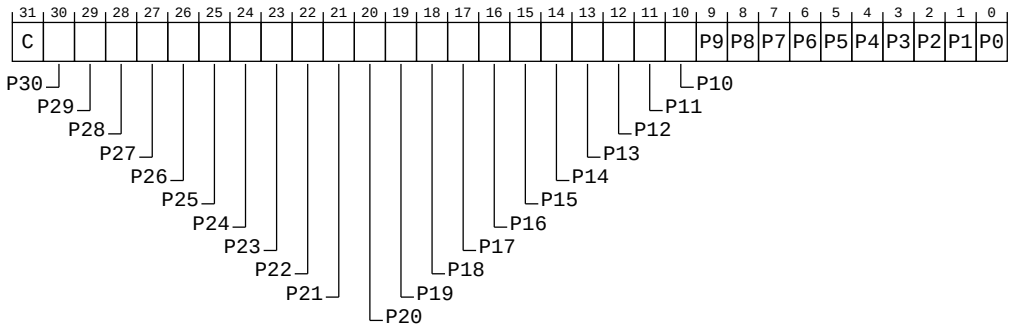


Table B-87: PMCNTENSET_EL0 bit descriptions

Bits	Name	Description	Reset
[31]	C	ext-PMCCNTR_EL0 enable bit. Enables the cycle counter register. Possible values are: **0b0** When read, means the cycle counter is disabled. When written, has no effect. **0b1** When read, means the cycle counter is enabled. When written, enables the cycle counter.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter enable bit for ext-PMEVCNTR<n>_EL0. If ext-PMCFGR.N is less than 31, bits [30:ext-PMCFGR.N] are **RAZ/WI**. **0b0** When read, means that ext-PMEVCNTR<n>_EL0 is disabled. When written, has no effect. **0b1** When read, means that ext-PMEVCNTR<n>_EL0 event counter is enabled. When written, enables ext-PMEVCNTR<n>_EL0.	31{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xC00	PMCNTENSET_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.33 PMCNTENCLR_EL0, Performance Monitors Count Enable Clear register

Disables the Cycle Count Register, ext-PMCCNTR_EL0, and any implemented event counters ext-PMEVCNTR<n>_EL0. Reading this register shows which counters are enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xC20

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-55: ext_pmcntenclr_el0 bit assignments

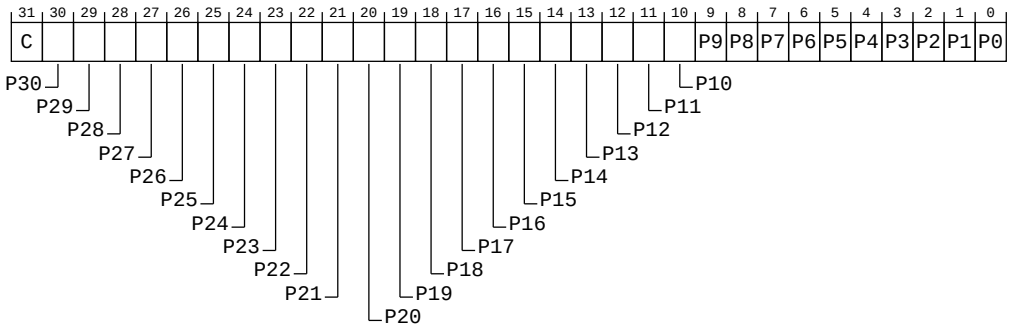


Table B-89: PMCNTENCLR_EL0 bit descriptions

Bits	Name	Description	Reset
[31]	C	ext-PMCCNTR_EL0 disable bit. Disables the cycle counter register. Possible values are: **0b0** When read, means the cycle counter is disabled. When written, has no effect. **0b1** When read, means the cycle counter is enabled. When written, disables the cycle counter.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter disable bit for ext-PMEVCNTR<n>_EL0. If ext-PMCFGR.N is less than 31, bits [30:ext-PMCFGR.N] are **RAZ/WI**. **0b0** When read, means that ext-PMEVCNTR<n>_EL0 is disabled. When written, has no effect. **0b1** When read, means that ext-PMEVCNTR<n>_EL0 is enabled. When written, disables ext-PMEVCNTR<n>_EL0.	31{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xC20	PMCNTENCLR_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.34 PMINTENSET_EL1, Performance Monitors Interrupt Enable Set register

Enables the generation of interrupt requests on overflows from the Cycle Count Register, ext-PMCCNTR_EL0, and the event counters ext-PMEVCNTR<n>_EL0. Reading the register shows which overflow interrupt requests are enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xC40

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-56: ext_pmintenset_el1 bit assignments

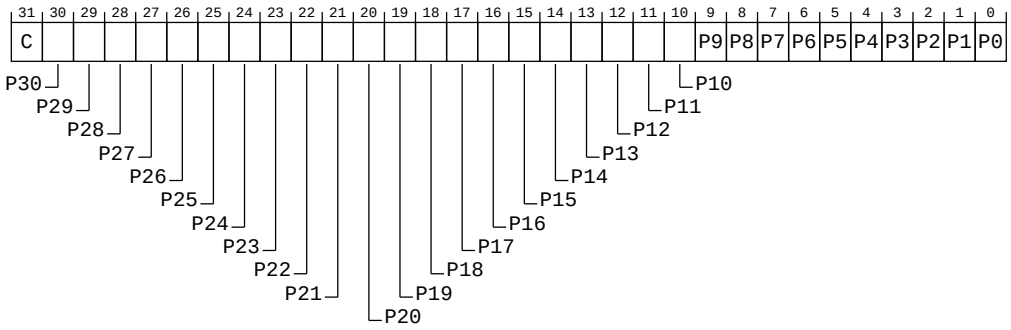


Table B-91: PMINTENSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	ext-PMCCNTR_EL0 overflow interrupt request enable bit. Possible values are: **0b0** When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect. **0b1** When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter overflow interrupt request enable bit for ext-PMEVCNTR<n>_EL0. If ext-PMCFGR.N is less than 31, bits [30:ext-PMCFGR.N] are **RAZ/WI**. **0b0** When read, means that the ext-PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect. **0b1** When read, means that the ext-PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, enables the ext-PMEVCNTR<n>_EL0 interrupt request.	31{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xC40	PMINTENSET_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.35 PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear register

Disables the generation of interrupt requests on overflows from the Cycle Count Register, ext-PMCCNTR_EL0, and the event counters ext-PMEVCNTR<n>_EL0. Reading the register shows which overflow interrupt requests are enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

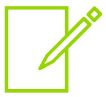
0xC60

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-57: ext_pmintenclr_el1 bit assignments

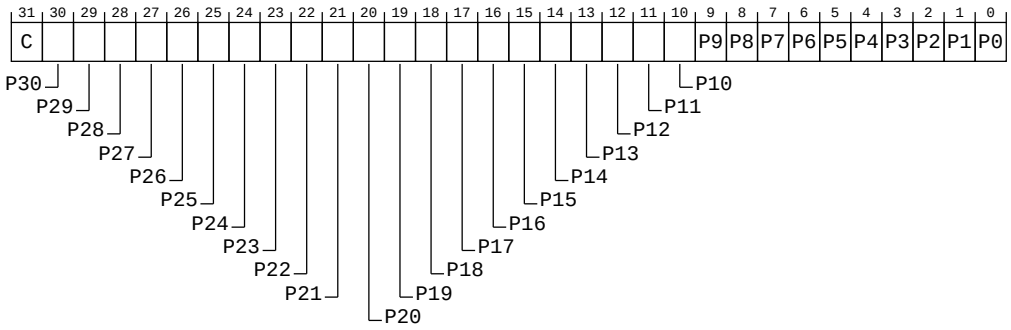


Table B-93: PMINTENCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	ext-PMCCNTR_EL0 overflow interrupt request disable bit. Possible values are: **0b0** When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect. **0b1** When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter overflow interrupt request disable bit for ext-PMEVCNTR<n>_EL0. If ext-PMCFGR.N is less than 31, bits [30:ext-PMCFGR.N] are **RAZ/WI**. **0b0** When read, means that the ext-PMEVCNTR<n>_EL0 event counter interrupt request is disabled. When written, has no effect. **0b1** When read, means that the ext-PMEVCNTR<n>_EL0 event counter interrupt request is enabled. When written, disables the ext-PMEVCNTR<n>_EL0 interrupt request.	31{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xC60	PMINTENCLR_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.36 PMOVSCLR_EL0, Performance Monitors Overflow Flag Status Clear register

Contains the state of the overflow bit for the Cycle Count Register, ext-PMCCNTR_EL0, and each of the implemented event counters ext-PMEVCNTR<n>_EL0. Writing to this register clears these bits.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xC80

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-58: ext_pmovsclr_el0 bit assignments

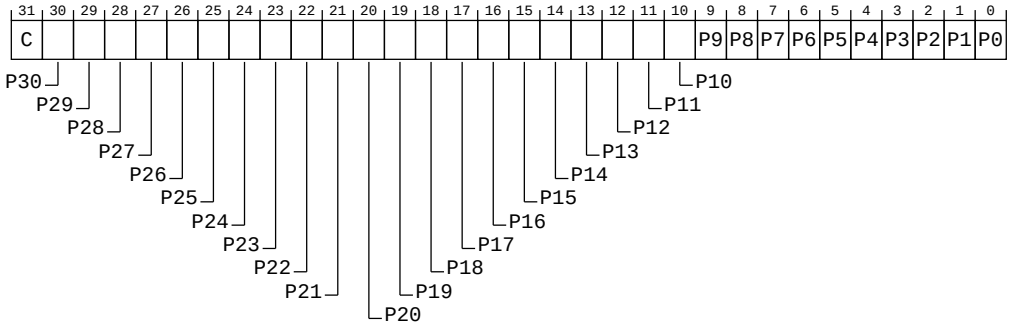


Table B-95: PMOVSCLR_EL0 bit descriptions

Bits	Name	Description	Reset
[31]	C	Cycle counter overflow clear bit. **0b0** When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect. **0b1** When read, means the cycle counter has overflowed since this bit was last cleared. When written, clears the cycle counter overflow bit to 0. ext-PMCR_EL0.LC controls whether an overflow is detected from unsigned overflow of ext-PMCCNTR_EL0[31:0] or unsigned overflow of ext-PMCCNTR_EL0[63:0].	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter overflow clear bit for ext-PMEVCNTR<n>_EL0. If ext-PMCFGR.N is less than 31, bits [30:ext-PMCFGR.N] are **RAZ/WI**. **0b0** When read, means that ext-PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect. **0b1** When read, means that ext-PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, clears the ext-PMEVCNTR<n>_EL0 overflow bit to 0. If FEAT_PMUv3p5 is implemented, AArch64-MDCR_EL2.HLP and ext-PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of ext-PMEVCNTR<n>_EL0[31:0] or unsigned overflow of ext-PMEVCNTR<n>_EL0[63:0].	31{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xC80	PMOVSCLR_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.37 PMSWINC_EL0, Performance Monitors Software Increment register

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see *SW_INCR* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If this register is implemented, use of it is deprecated.

If 1 is written to bit [n] from the external debug interface, it is CONSTRAINED UNPREDICTABLE whether or not a SW_INCR event is created for counter n. This is consistent with not implementing the register in the external debug interface.

Attributes

Width

32

Component

PMU

Register offset

0xCA0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-59: ext_pmswinc_el0 bit assignments

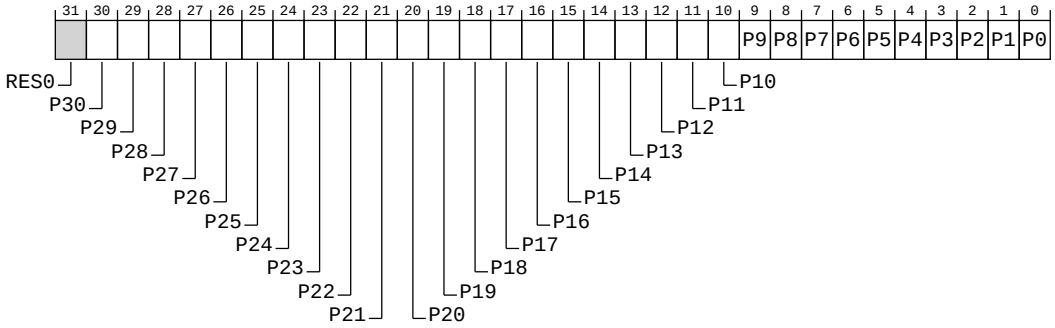


Table B-97: PMSWINC_EL0 bit descriptions

Bits	Name	Description	Reset
[31]	**RES0**	Reserved	**RES0**
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter software increment bit for ext-PMEVCNTR<n>_EL0. If ext-PMCFGR.N is less than 31, bits [30:ext-PMCFGR.N] are **WI**. **0b0** No action. The write to this bit is ignored. **0b1** It is **CONSTRAINED UNPREDICTABLE** whether a SW_INCR event is generated for event counter n.	`31{x}`

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xCA0	PMSWINC_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

WI

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

WO

Otherwise

ERROR

B.3.38 PMOVSSET_EL0, Performance Monitors Overflow Flag Status Set register

Sets the state of the overflow bit for the Cycle Count Register, ext-PMCCNTR_EL0, and each of the implemented event counters ext-PMEVCNTR<n>_EL0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xCC0

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-60: ext_pmovsset_el0 bit assignments

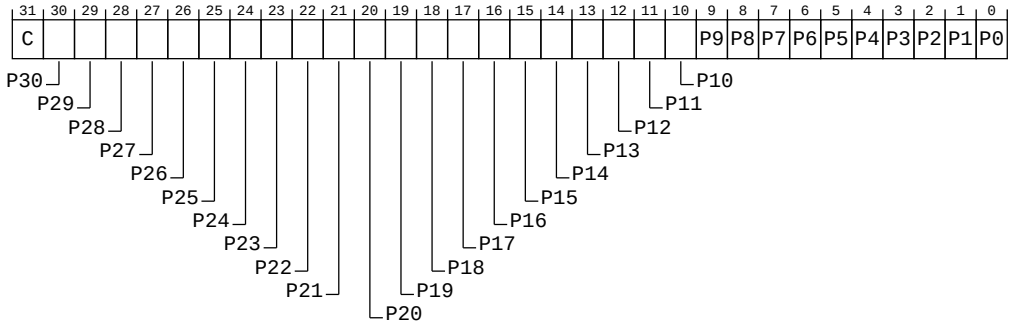


Table B-99: PMOVSSET_EL0 bit descriptions

Bits	Name	Description	Reset
[31]	C	Cycle counter overflow set bit. **0b0** When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect. **0b1** When read, means the cycle counter has overflowed since this bit was last cleared. When written, sets the cycle counter overflow bit to 1.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter overflow set bit for ext-PMEVCNTR<n>_EL0. If ext-PMCFGR.N is less than 31, bits [30:ext-PMCFGR.N] are **RAZ/WI**. **0b0** When read, means that ext-PMEVCNTR<n>_EL0 has not overflowed since this bit was last cleared. When written, has no effect. **0b1** When read, means that ext-PMEVCNTR<n>_EL0 has overflowed since this bit was last cleared. When written, sets the ext-PMEVCNTR<n>_EL0 overflow bit to 1. If FEAT_PMUv3p5 is implemented, AArch64-MDCR_EL2.HLP and ext-PMCR_EL0.LP control whether an overflow is detected from unsigned overflow of ext-PMEVCNTR<n>_EL0[31:0] or unsigned overflow of ext-PMEVCNTR<n>_EL0[63:0].	31{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xCC0	PMOVSSET_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.39 PMCFGR, Performance Monitors Configuration Register

Contains PMU-specific configuration data.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE00

Access type

See bit descriptions

Reset value

0000 xxxx xxxx 0001 1111 1111 xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-61: ext_pmcfgr bit assignments

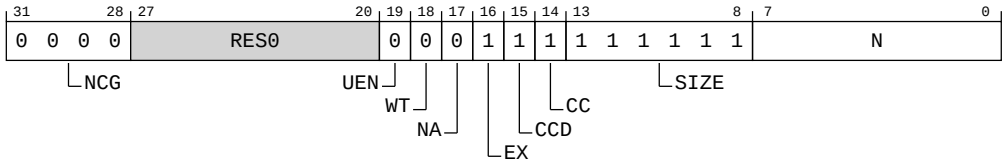


Table B-101: PMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	This feature is not supported, so this field is **RAZ**. **0b0000**	0b0000
[27:20]	**RES0**	Reserved	**RES0**
[19]	UEN	User-mode Enable Register supported. AArch64-PMUSERENR_EL0 is not visible in the external debug interface, so this bit is **RAZ**. **0b0**	0b0

Bits	Name	Description	Reset
[18]	WT	This feature is not supported, so this bit is **RAZ**. **0b0**	0b0
[17]	NA	This feature is not supported, so this bit is **RAZ**. **0b0**	0b0
[16]	EX	Export supported. **0b1** ext-PMCR_EL0.X is read/write.	0b1
[15]	CCD	Cycle counter has prescale. **0b1** ext-PMCR_EL0.D is read/write.	0b1
[14]	CC	Dedicated cycle counter (counter 31) supported. **0b1**	0b1
[13:8]	SIZE	Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit. From Armv8, the largest counter is 64-bits, so the value of this field is 0b111111. This field is used by software to determine the spacing of the counters in the memory-map. From Armv8, the counters are a doubleword-aligned addresses. **0b111111**	0b111111
[7:0]	N	Number of counters implemented in addition to the cycle counter, ext-PMCCNTR_EL0. **0b00000110** 6 PMU counters **0b00010100** 20 PMU counters Must be configured to either 0x06 or 0x14	The reset values can be the following: 0b00000110, 0b00010100, respective to the value.

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE00	PMCFGR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.3.40 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

32

Component

PMU

Register offset

0xE04

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0xxx xxx0 x000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-62: ext_pmcr_el0 bit assignments

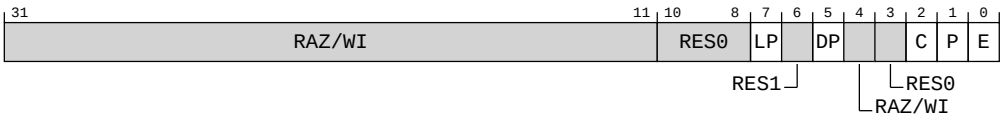


Table B-103: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[31:11]	**RAZ/WI**	Reserved	**RAZ/WI**
[10:8]	**RES0**	Reserved	**RES0**
[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].	x
[6]	**RES1**	Reserved	**RES1**
[5]	DP	Disable cycle counter when event counting is prohibited. The possible values of this bit are: **0b0** Cycle counting by ext-PMCCNTR_EL0 is not affected by this mechanism. **0b1** Cycle counting by ext-PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen: • If FEAT_PMUv3p1 is implemented, EL2 is implemented, and AArch64-MDCR_EL2.HPMD is 1, then cycle counting by ext-PMCCNTR_EL0 is disabled at EL2. • If EL3 is implemented, AArch64-MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or AArch64-MDCR_EL3.MPMX is 0, then cycle counting by ext-PMCCNTR_EL0 is disabled at EL3 and in Secure state. If AArch64-MDCR_EL2.HPMN is not 0, this is when event counting by event counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited or frozen.	x
[4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3]	**RES0**	Reserved	**RES0**
[2]	C	Cycle counter reset. The effects of writing to this bit are: **0b1** Reset ext-PMCCNTR_EL0 to zero. Access to this field is: WO/**RAZ**	0b0
[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. Access to this field is: WO/**RAZ**	0b0
[0]	E	Enable **0b1** ext-PMCCNTR_EL0 and event counters ext-PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are enabled by ext-PMCNTENSET_EL0.	0b0

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE04	PMCR_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.3.41 PMCEID0, Performance Monitors Common Event Identification register 0

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.


Note

This view of the register was previously called PMCEID0_EL0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE20

Access type

See bit descriptions

Reset value

0111 1111 1111 1111 0000 1111 0011 1111

Bit descriptions

Figure B-63: ext_pmceid0 bit assignments

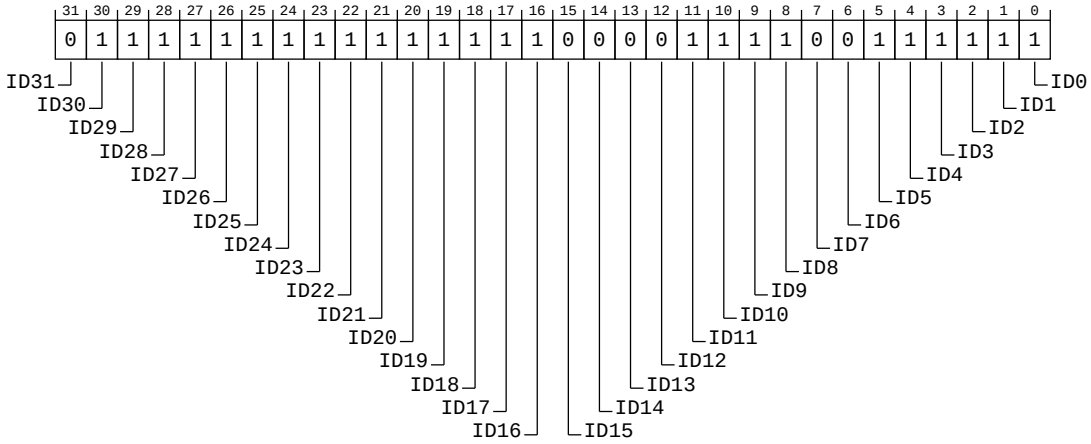


Table B-105: PMCEID0 bit descriptions

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.	0b0
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN **0b1** The Common event is implemented.	0b1
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES **0b1** The Common event is implemented.	0b1
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED **0b1** The Common event is implemented.	0b1
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC **0b1** The Common event is implemented.	0b1
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS **0b1** The Common event is implemented.	0b1
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB **0b1** The Common event is implemented.	0b1
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0b1** The Common event is implemented.	0b1
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB **0b1** The Common event is implemented.	0b1
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE **0b1** The Common event is implemented.	0b1
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS **0b1** The Common event is implemented.	0b1
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED **0b1** The Common event is implemented.	0b1
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES **0b1** The Common event is implemented.	0b1
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED **0b1** The Common event is implemented.	0b1
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **0b1** The Common event is implemented.	0b1
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **0b1** The Common event is implemented.	0b1
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN **0b1** The Common event is implemented.	0b1
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED **0b1** The Common event is implemented.	0b1
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **0b0** The Common event is not implemented, or not counted.	0b0
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL **0b1** The Common event is implemented.	0b1
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE **0b1** The Common event is implemented.	0b1
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL **0b1** The Common event is implemented.	0b1
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR **0b1** The Common event is implemented.	0b1

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE20	PMCEID0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.3.42 PMCEID1, Performance Monitors Common Event Identification register 1

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.


Note

This view of the register was previously called PMCEID1_EL0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE24

Access type

See bit descriptions

Reset value

1111 1110 1111 0010 1010 1110 0111 1111

Bit descriptions

Figure B-64: ext_pmceid1 bit assignments

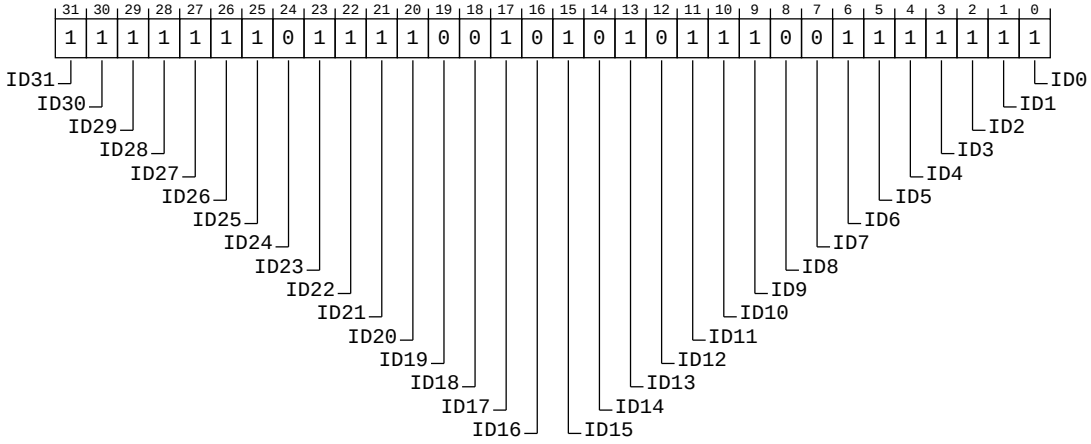


Table B-107: PMCEID1 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **0b1** The Common event is implemented.	0b1
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **0b1** The Common event is implemented.	0b1
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **0b1** The Common event is implemented.	0b1
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **0b1** The Common event is implemented.	0b1
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **0b1** The Common event is implemented.	0b1
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **0b1** The Common event is implemented.	0b1
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD **0b0** The Common event is not implemented, or not counted.	0b0
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD **0b1** The Common event is implemented.	0b1
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD **0b1** The Common event is implemented.	0b1
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WLK **0b1** The Common event is implemented.	0b1
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WLK **0b1** The Common event is implemented.	0b1
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) **0b0** The Common event is not implemented, or not counted.	0b0
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) **0b0** The Common event is not implemented, or not counted.	0b0
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS **0b1** The Common event is implemented.	0b1
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB **0b0** The Common event is not implemented, or not counted.	0b0
[15]	ID15	ID15 corresponds to common event (0x2f) L2TLB_REQ **0b1** The Common event is implemented.	0b1
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **0b0** The Common event is not implemented, or not counted.	0b0
[13]	ID13	ID13 corresponds to common event (0x2d) L2TLB_REFILL **0b1** The Common event is implemented.	0b1
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0b1** The Common event is implemented.	0b1
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0b1** The Common event is implemented.	0b1
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL **0b0** The Common event is not implemented, or not counted.	0b0
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE **0b0** The Common event is not implemented, or not counted.	0b0
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB **0b1** The Common event is implemented.	0b1
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB **0b1** The Common event is implemented.	0b1
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND **0b1** The Common event is implemented.	0b1
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND **0b1** The Common event is implemented.	0b1
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED **0b1** The Common event is implemented.	0b1
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED **0b1** The Common event is implemented.	0b1
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE **0b1** The Common event is implemented.	0b1

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE24	PMCEID1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.3.43 PMCEID2, Performance Monitors Common Event Identification register 2

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE28

Access type

See bit descriptions

Reset value

0000 1111 0000 1111 0001 1010 0111 1111

Bit descriptions

Figure B-65: ext_pmceid2 bit assignments

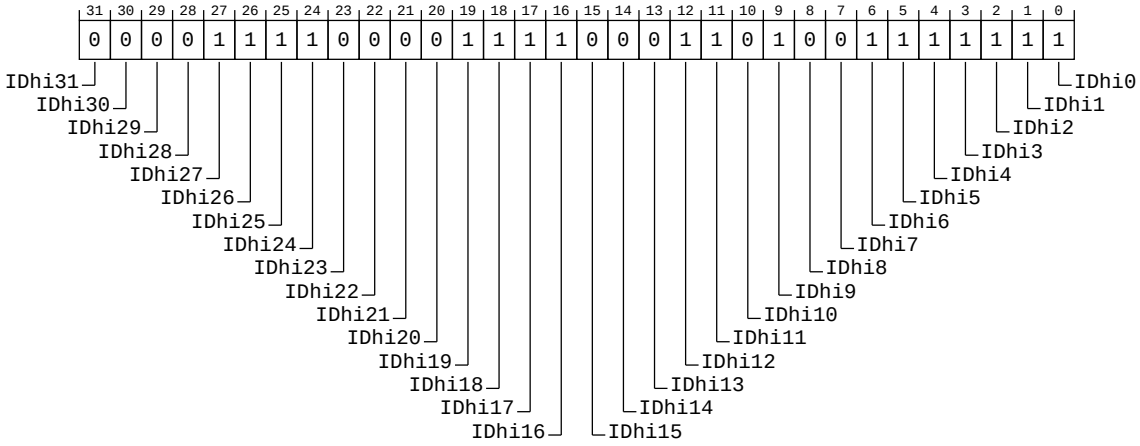


Table B-109: PMCEID2 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) **0b0** The Common event is not implemented, or not counted.	0b0
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) **0b0** The Common event is not implemented, or not counted.	0b0
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) **0b0** The Common event is not implemented, or not counted.	0b0
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) **0b0** The Common event is not implemented, or not counted.	0b0
[27]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 **0b1** The Common event is implemented.	0b1
[26]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 **0b1** The Common event is implemented.	0b1
[25]	IDhi25	IDhi25 corresponds to common event (0x4019) CTI_TRIGOUT5 **0b1** The Common event is implemented.	0b1
[24]	IDhi24	IDhi24 corresponds to common event (0x4018) CTI_TRIGOUT4 **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4017) **0b0** The Common event is not implemented, or not counted.	0b0
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4016) **0b0** The Common event is not implemented, or not counted.	0b0
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4015) **0b0** The Common event is not implemented, or not counted.	0b0
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4014) **0b0** The Common event is not implemented, or not counted.	0b0
[19]	IDhi19	IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 **0b1** The Common event is implemented.	0b1
[18]	IDhi18	IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 **0b1** The Common event is implemented.	0b1
[17]	IDhi17	IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 **0b1** The Common event is implemented.	0b1
[16]	IDhi16	IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 **0b1** The Common event is implemented.	0b1
[15]	IDhi15	IDhi15 corresponds to common event (0x400f) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[14]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG **0b0** The Common event is not implemented, or not counted.	0b0
[13]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS **0b0** The Common event is not implemented, or not counted.	0b0
[12]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP **0b1** The Common event is implemented.	0b1
[11]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0b1** The Common event is implemented.	0b1
[10]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[9]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0b1** The Common event is implemented.	0b1
[8]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[7]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[6]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS **0b1** The Common event is implemented.	0b1
[5]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM **0b1** The Common event is implemented.	0b1
[4]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0b1** The Common event is implemented.	0b1
[3]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0b1** The Common event is implemented.	0b1
[2]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0b1** The Common event is implemented.	0b1
[1]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0b1** The Common event is implemented.	0b1
[0]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0b1** The Common event is implemented.	0b1

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE28	PMCEID2	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.3.44 PMCEID3, Performance Monitors Common Event Identification register 3

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE2C

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0111 0111

Bit descriptions

Figure B-66: ext_pmceid3 bit assignments

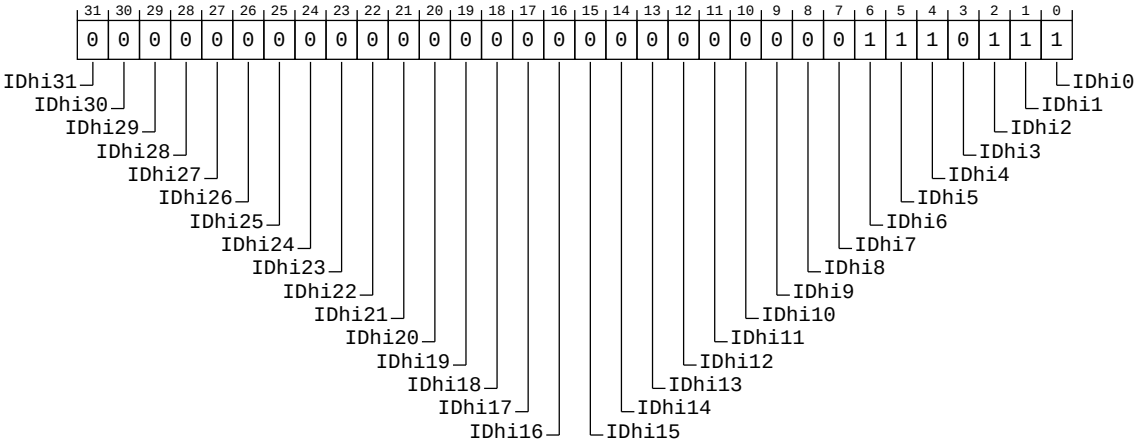


Table B-111: PMCEID3 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) **0b0** The Common event is not implemented, or not counted.	0b0
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) **0b0** The Common event is not implemented, or not counted.	0b0
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) **0b0** The Common event is not implemented, or not counted.	0b0
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) **0b0** The Common event is not implemented, or not counted.	0b0
[27]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) **0b0** The Common event is not implemented, or not counted.	0b0
[26]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) **0b0** The Common event is not implemented, or not counted.	0b0
[25]	IDhi25	IDhi25 corresponds to a Reserved Event event (0x4039) **0b0** The Common event is not implemented, or not counted.	0b0
[24]	IDhi24	IDhi24 corresponds to a Reserved Event event (0x4038) **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) **0b0** The Common event is not implemented, or not counted.	0b0
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) **0b0** The Common event is not implemented, or not counted.	0b0
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) **0b0** The Common event is not implemented, or not counted.	0b0
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) **0b0** The Common event is not implemented, or not counted.	0b0
[19]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) **0b0** The Common event is not implemented, or not counted.	0b0
[18]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) **0b0** The Common event is not implemented, or not counted.	0b0
[17]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) **0b0** The Common event is not implemented, or not counted.	0b0
[16]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) **0b0** The Common event is not implemented, or not counted.	0b0
[15]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **0b0** The Common event is not implemented, or not counted.	0b0
[14]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **0b0** The Common event is not implemented, or not counted.	0b0
[13]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **0b0** The Common event is not implemented, or not counted.	0b0
[12]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **0b0** The Common event is not implemented, or not counted.	0b0
[11]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **0b0** The Common event is not implemented, or not counted.	0b0
[10]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **0b0** The Common event is not implemented, or not counted.	0b0

Bits	Name	Description	Reset
[9]	IDhi9	IDhi9 corresponds to a Reserved Event event (0x4029) **0b0** The Common event is not implemented, or not counted.	0b0
[8]	IDhi8	IDhi8 corresponds to a Reserved Event event (0x4028) **0b0** The Common event is not implemented, or not counted.	0b0
[7]	IDhi7	IDhi7 corresponds to a Reserved Event event (0x4027) **0b0** The Common event is not implemented, or not counted.	0b0
[6]	IDhi6	IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR **0b1** The Common event is implemented.	0b1
[5]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD **0b1** The Common event is implemented.	0b1
[4]	IDhi4	IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED **0b1** The Common event is implemented.	0b1
[3]	IDhi3	IDhi3 corresponds to common event (0x4023) Reserved **0b0** The Common event is not implemented, or not counted.	0b0
[2]	IDhi2	IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT **0b1** The Common event is implemented.	0b1
[1]	IDhi1	IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT **0b1** The Common event is implemented.	0b1
[0]	IDhi0	IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT **0b1** The Common event is implemented.	0b1

Access

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
PMU	0xE2C	PMCEID3	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.3.45 PMMIR, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0xE40

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-67: ext_pmmir bit assignments

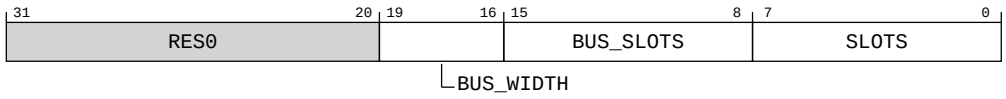


Table B-113: PMMIR bit descriptions

Bits	Name	Description	Reset
[31:20]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[19:16]	BUS_WIDTH	Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as Log_2(number of bytes), plus one. **0b0000** The information is not available. **0b0011** Four bytes. **0b0100** 8 bytes. **0b0101** 16 bytes. **0b0110** 32 bytes. **0b0111** 64 bytes. **0b1000** 128 bytes. **0b1001** 256 bytes. **0b1010** 512 bytes. **0b1011** 1024 bytes. **0b1100** 2048 bytes. All other values are reserved. Each transfer is up to this number of bytes. An access might be smaller than the bus width. When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.	The reset values can be the following: `0b0000`, `0b0011`, `0b0100`, `0b0101`, `0b0110`, `0b0111`, `0b1000`, `0b1001`, `0b1010`, `0b1011`, `0b1100`, respective to the value.
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS. If the information is not available, this field will read as zero.	`8{x}`
[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.	`8{x}`

Accessibility

If the Core power domain is off or in a low-power state, access on this interface returns an Error.

Component	Offset	Instance	Range
PMU	0xE40	PMMIR	None

This interface is accessible as follows:

When !IsCorePowered() || DoubleLockStatus() || OSLockStatus() || !AllowExternalPMUAccess()

ERROR

Otherwise

RO

B.3.46 PMDEVAFF0, Performance Monitors Device Affinity register 0

Copy of the low half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required if the external interface to the PMU is implemented.

Attributes

Width

32

Component

PMU

Register offset

0xFA8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-68: ext_pmdevaff0 bit assignments

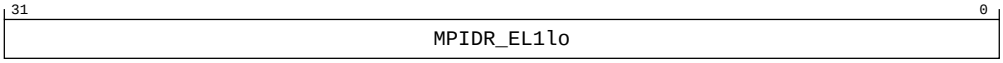


Table B-115: PMDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1lo	AArch64-MPIDR_EL1 low half. Read-only copy of the low half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32{x}

Accessibility

Component	Offset	Instance	Range
PMU	0xFA8	PMDEVAFF0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.47 PMDEVAFF1, Performance Monitors Device Affinity register 1

Copy of the high half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required if the external interface to the PMU is implemented.

Attributes

Width

32

Component

PMU

Register offset

0xFAC

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-69: ext_pmdevaff1 bit assignments

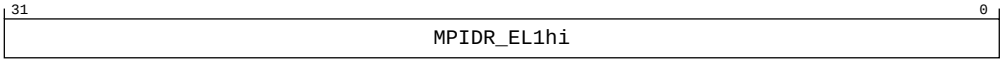


Table B-117: PMDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1hi	AArch64-MPIDR_EL1 high half. Read-only copy of the high half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32{x}

Accessibility

Component	Offset	Instance	Range
PMU	0xFAC	PMDEVAFF1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.48 PMLAR, Performance Monitors Lock Access Register

Allows or disallows access to the Performance Monitors registers through a memory-mapped interface.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

If FEAT_DoPD is implemented, Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Software uses PMLAR to set or clear the lock, and ext-PMLSR to check the current status of the lock.

Attributes

Width

32

Component

PMU

Register offset

0xFB0

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-70: ext_pmlar bit assignments

Table B-119: PMLAR bit descriptions

Bits	Name	Description	Reset
[31:0]	KEY	Lock Access control. Writing the key value `0xC5ACCE55` to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.	`32{x}`

Accessibility

Component	Offset	Instance	Range
PMU	0xFB0	PMLAR	None

This interface is accessible as follows:

When IsCorePowered()

WO

Otherwise

ERROR

B.3.49 PMLSR, Performance Monitors Lock Status Register

Indicates the current status of the software lock for Performance Monitors registers.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

If FEAT_DoPD is implemented, Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Software uses ext-PMLAR to set or clear the lock, and PMLSR to check the current status of the lock.

Attributes

Width

32

Component

PMU

Register offset

0xFB4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x00x

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-71: ext_pmlsr bit assignments

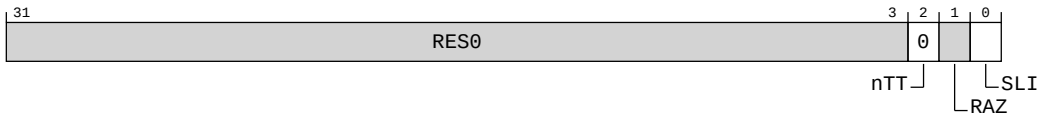


Table B-121: PMLSR bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2]	nTT	Not thirty-two bit access required. **0b0**	0b0
[1]	**RAZ**	Reserved	**RAZ**
[0]	SLI	Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is **IMPLEMENTATION DEFINED**. Permitted values are: **0b0** Software Lock not implemented or not memory-mapped access. **0b1** Software Lock implemented and memory-mapped access.	x

Accessibility

Component	Offset	Instance	Range
PMU	0xFB4	PMLSR	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.3.50 PMAUTHSTATUS, Performance Monitors Authentication Status register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for Performance Monitors.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is OPTIONAL, and is required for CoreSight compliance. Arm recommends that this register is implemented.

Attributes

Width

32

Component

PMU

Register offset

0xFB8

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-72: ext_pmauthstatus bit assignments

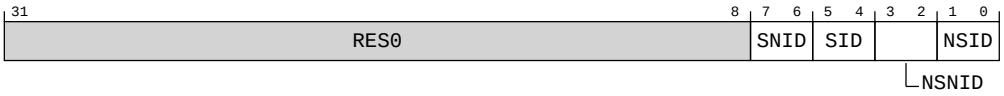


Table B-123: PMAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:6]	SNID	Holds the same value as ext-DBGAUTHSTATUS_EL1.SNID.	xx
[5:4]	SID	Secure invasive debug. Possible values of this field are: **0b00** Not implemented. All other values are reserved.	xx
[3:2]	NSNID	Holds the same value as ext-DBGAUTHSTATUS_EL1.NSNID.	xx
[1:0]	NSID	Non-secure invasive debug. Possible values of this field are: **0b00** Not implemented. All other values are reserved.	xx

Accessibility

Component	Offset	Instance	Range
PMU	0xFB8	PMAUTHSTATUS	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.51 PMDEVARCH, Performance Monitors Device Architecture register

Identifies the programmers' model architecture of the Performance Monitor component.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFBC

Access type

See bit descriptions

Reset value

0100 0111 0111 0000 0010 1010 0001 0110

Bit descriptions

Figure B-73: ext_pmdevarch bit assignments

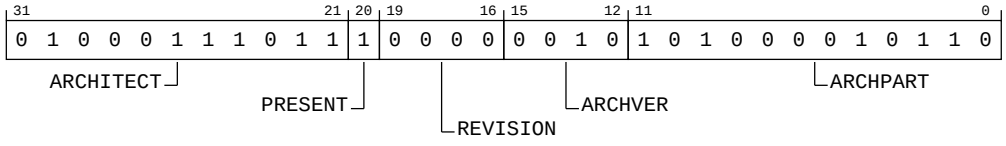


Table B-125: PMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For Performance Monitors, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0x4`. Bits [27:21] are the JEP106 ID code, `0x3B`. **`0b01000111011`**	`0b01000111011`
[20]	PRESENT	Indicates that the DEVARCH is present. **`0b1`**	`0b1`
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For Performance Monitors, the revision defined by Armv8 is `0x0`. All other values are reserved. **`0b0000`**	`0b0000`
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **`0b0010`** Performance Monitors Extension version 3, PMUv3. All other values are reserved. PMDEVARCH.ARCHVER and PMDEVARCH.ARCHPART are also defined as a single field, PMDEVARCH.ARCHID, so that PMDEVARCH.ARCHVER is PMDEVARCH.ARCHID[15:12].	`0b0010`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **`0b101000010110`** Armv8-A PE performance monitors. PMDEVARCH.ARCHVER and PMDEVARCH.ARCHPART are also defined as a single field, PMDEVARCH.ARCHID, so that PMDEVARCH.ARCHPART is PMDEVARCH.ARCHID[11:0].	`0xA16`

Accessibility

Component	Offset	Instance	Range
PMU	0xFBC	PMDEVARCH	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.52 PMDEVID, Performance Monitors Device ID register

Provides information about features of the Performance Monitors implementation.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required from Armv8.2 and in any implementation that includes FEAT_PCSRv8p2. Otherwise, its location is RES0.

Before Armv8.2, the PC Sample-based Profiling Extension can be implemented in the external debug register space, as indicated by the value of ext-EDDEVID.PCSample.

Attributes

Width

32

Component

PMU

Register offset

0xFC8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0001

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-74: ext_pmdevid bit assignments

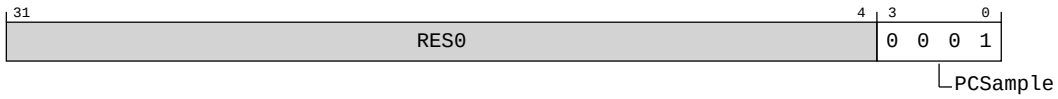


Table B-127: PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. **0b0001** PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	0b0001

Accessibility

Component	Offset	Instance	Range
PMU	0xFC8	PMDEVID	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.53 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PEs performance monitor interface.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFCC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-75: ext_pmdevtype bit assignments

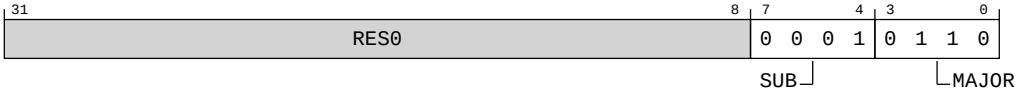


Table B-129: PMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Indicates this is a component within a PE. **0b0001**	0b0001
[3:0]	MAJOR	Major type. Indicates this is a performance monitor component. **0b0110**	0b0110

Accessibility

Component	Offset	Instance	Range
PMU	0xFCC	PMDEVTYPE	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.54 PMPIDR4, Performance Monitors Peripheral Identification Register 4

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFD0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-76: ext_pmpidr4 bit assignments

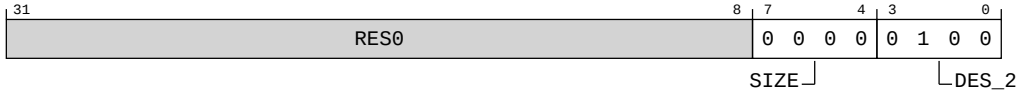


Table B-131: PMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	4KB count. **0b0000**	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited. This is bits[3:0] of the JEP106 continuation code.	0b0100

Accessibility

Component	Offset	Instance	Range
PMU	0xFD0	PMPIDR4	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.55 PMPIDR0, Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0100 1110

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-77: ext_pmpidr0 bit assignments

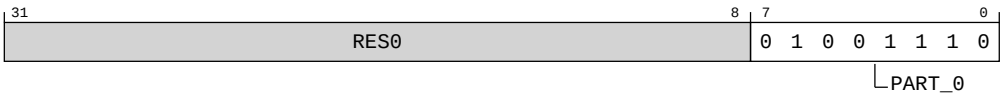


Table B-133: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b01001110** Least significant byte of the PMU unit part.	0x4E

Accessibility

Component	Offset	Instance	Range
PMU	0xFE0	PMPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.56 PMPIDR1, Performance Monitors Peripheral Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-78: ext_pmpidr1 bit assignments

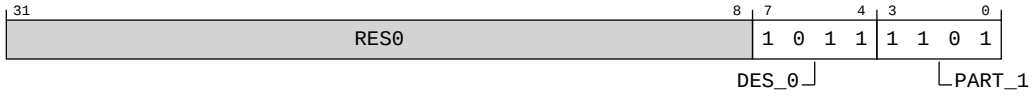


Table B-135: PMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited. This is the least significant nibble of JEP106 ID code.	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** Part number, most significant nibble.	`0b1101`

Accessibility

Component	Offset	Instance	Range
PMU	0xFE4	PMPIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.57 PMPIDR2, Performance Monitors Peripheral Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFE8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 1011

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-79: ext_pmpidr2 bit assignments

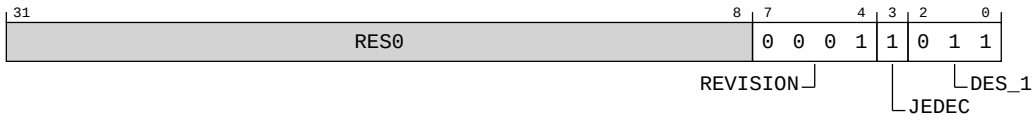


Table B-137: PMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited. This is bits[6:4] of the JEP106 ID code.	0b011

Accessibility

Component	Offset	Instance	Range
PMU	0xFE8	PMPIDR2	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.3.58 PMPIDR3, Performance Monitors Peripheral Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width
32

Component
PMU

Register offset

0xFEC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-80: ext_pmpidr3 bit assignments

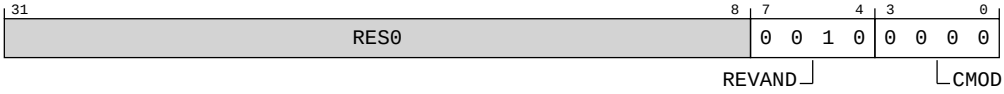


Table B-139: PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using ext-PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010**	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000** The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFEC	PMPIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.59 PMCIDR0, Performance Monitors Component Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-81: ext_pmcidr0 bit assignments

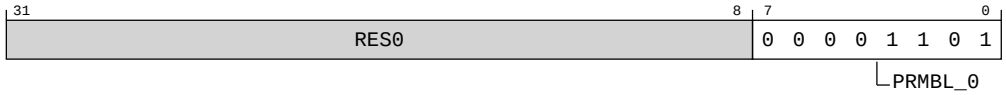


Table B-141: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PRMBL_0	Preamble. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
PMU	0xFF0	PMCIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.60 PMCIDR1, Performance Monitors Component Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-82: ext_pmcidr1 bit assignments

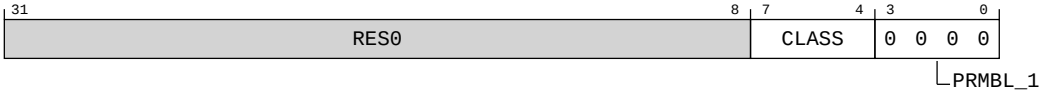


Table B-143: PMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component.	xxxx
[3:0]	PRMBL_1	Preamble. **RAZ**. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFF4	PMCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.61 PMCIDR2, Performance Monitors Component Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFF8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-83: ext_pmcidr2 bit assignments

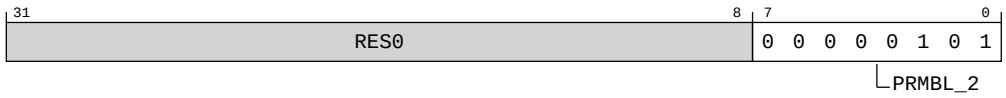


Table B-145: PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0b00000101**	0x05

Accessibility

Component	Offset	Instance	Range
PMU	0xFF8	PMCIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.3.62 PMCIDR3, Performance Monitors Component Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

PMU

Register offset

0xFFC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-84: ext_pmcidr3 bit assignments

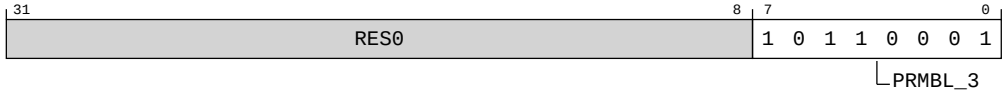


Table B-147: PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
PMU	0xFFC	PMCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4 External CTI registers summary

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

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

Table B-149: CTI registers summary

Offset	Name	Reset	Width	Description
0x000	CTICONTROL	–	32-bit	CTI Control register
0x010	CTIINTACK	–	32-bit	CTI Output Trigger Acknowledge register
0x014	CTIAPPSET	–	32-bit	CTI Application Trigger Set register
0x018	CTIAPPCLEAR	–	32-bit	CTI Application Trigger Clear register
0x01C	CTIAPPPULSE	–	32-bit	CTI Application Pulse register
0x020 + (4 * n)	CTIINEN_n_	–	32-bit	CTI Input Trigger to Output Channel Enable registers
0x0A0 + (4 * n)	CTIOUTEN_n_	–	32-bit	CTI Input Channel to Output Trigger Enable registers
0x130	CTITRIGINSTATUS	–	32-bit	CTI Trigger In Status register
0x134	CTITRIGOUTSTATUS	–	32-bit	CTI Trigger Out Status register
0x138	CTICHINSTATUS	–	32-bit	CTI Channel In Status register
0x13C	CTICHOUTSTATUS	–	32-bit	CTI Channel Out Status register
0x140	CTIGATE	–	32-bit	CTI Channel Gate Enable register
0x144	ASICCTL	–	32-bit	CTI External Multiplexer Control register
0x150	CTIDEVCTL	–	32-bit	CTI Device Control register
0xFA8	CTIDEVAFF0	–	32-bit	CTI Device Affinity register 0

Offset	Name	Reset	Width	Description
0xFAC	CTIDEVAFF1	—	32-bit	CTI Device Affinity register 1
0xFB0	CTILAR	—	32-bit	CTI Lock Access Register
0xFB4	CTILSR	—	32-bit	CTI Lock Status Register
0xFB8	CTIAUTHSTATUS	—	32-bit	CTI Authentication Status register
0xFBC	CTIDEVARCH	—	32-bit	CTI Device Architecture register
0xFC0	CTIDEVID2	—	32-bit	CTI Device ID register 2
0xFC4	CTIDEVID1	—	32-bit	CTI Device ID register 1
0xFC8	CTIDEVID	—	32-bit	CTI Device ID register 0

B.4.1 CTICONTROL, CTI Control register

Controls whether the CTI is enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x000

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-85: ext_cticontrol bit assignments

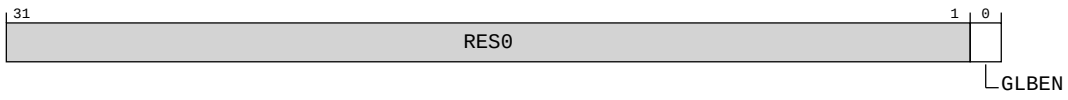


Table B-150: CTICONTROL bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	GLBEN	Enables or disables the CTI mapping functions. Possible values of this field are: **0b0** CTI mapping functions and application trigger disabled. **0b1** CTI mapping functions and application trigger enabled. When GLBEN is 0, the input channel to output trigger, input trigger to output channel, and application trigger functions are disabled and do not signal new events on either output triggers or output channels. If a previously asserted output trigger has not been acknowledged, it is **CONSTRAINED UNPREDICTABLE** which of the following occurs: • The output trigger remains asserted after the mapping functions are disabled. • The output trigger is deasserted after the mapping functions are disabled. All output triggers are disabled by CTI reset. If the ECT supports multicycle channel events any existing output channel events will be terminated.	0b0

Accessibility

Component	Offset	Instance	Range
CTI	0x000	CTICONTROL	None

This interface is accessible as follows:

When SoftwareLockStatus()

RO

When !SoftwareLockStatus()

RW

B.4.2 CTIINTACK, CTI Output Trigger Acknowledge register

Can be used to deactivate the output triggers.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x010

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-86: ext_ctiintack bit assignments

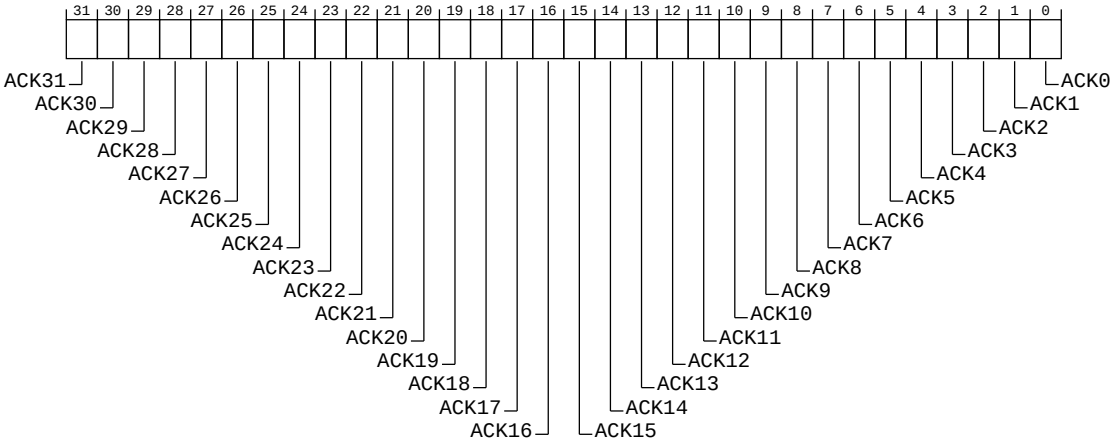


Table B-152: CTIINTACK bit descriptions

Bits	Name	Description	Reset
[31:0]	ACK<n>, bit[n], where n = 31 to 0	Acknowledge for output trigger <n>. Bits [31:N] are RAZ/WI. N is the number of CTI triggers implemented as defined by the ext-CTIDEVID.NUMTRIG field. If any of the following is true, writes to ACK<n> are ignored: • n >= ext-CTIDEVID.NUMTRIG, the number of implemented triggers. • Output trigger n is not active. • The channel mapping function output, as controlled by ext-CTIOUTEN<n>, is still active. Otherwise, if any of the following are true, ACK<n> is **RES0**: • Output trigger n is not implemented. • Output trigger n is not connected. • Output trigger n is self-acknowledging and does not require software acknowledge. Otherwise, the behavior on writes to ACK<n> is as follows: **0b0** No effect **0b1** Deactivate the trigger.	32{x}

Accessibility

A debugger must read ext-CTITRIGOUTSTATUS to confirm that the output trigger has been acknowledged before generating any event that must be ordered after the write to CTIINTACK, such as a write to CTIAPPPULSE to activate another trigger.

Component	Offset	Instance	Range
CTI	0x010	CTIINTACK	None

This interface is accessible as follows:

When SoftwareLockStatus()
WI

When !SoftwareLockStatus()
WO

B.4.3 CTIAPPSET, CTI Application Trigger Set register

Sets the application triggers.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x014

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-87: ext_ctiappset bit assignments

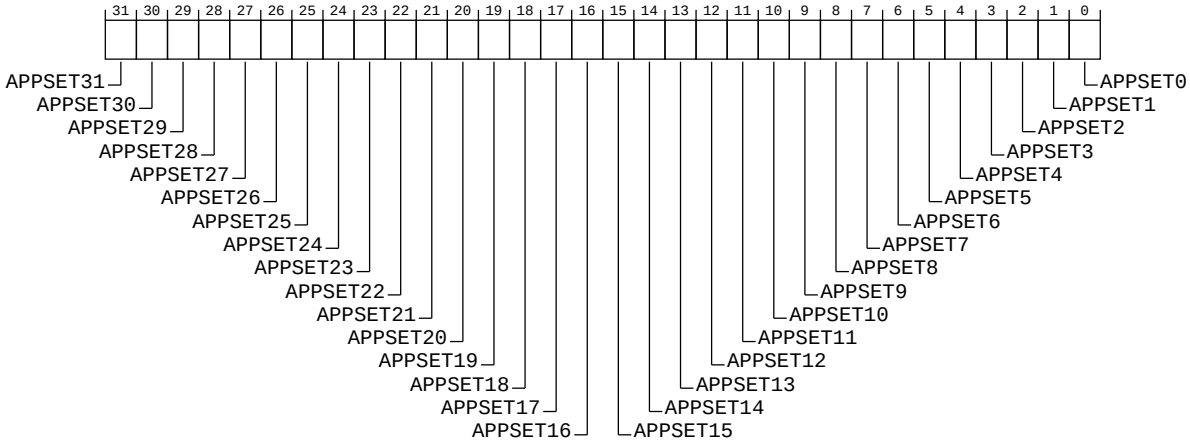


Table B-154: CTIAPPSET bit descriptions

Bits	Name	Description	Reset
[31:0]	APPSET<x>, bit[x], where x = 31 to 0	Application trigger <x> enable. Bits [31:N] are **RAZ/WI**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. **0b0** Reading this means the application trigger is inactive. Writing this has no effect. **0b1** Reading this means the application trigger is active. Writing this sets the corresponding application trigger to 1 and generates a channel event. If the ECT does not support multicycle channel events, use of CTIAPPSET is deprecated and the debugger must only use ext-CTIAPPPULSE.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x014	CTIAPPSET	None

This interface is accessible as follows:

When SoftwareLockStatus()
RO

When !SoftwareLockStatus()
RW

B.4.4 CTIAPPCLEAR, CTI Application Trigger Clear register

Clears the application triggers.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
CTI

Register offset
0x018

Access type
See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-88: ext_ctiappclear bit assignments

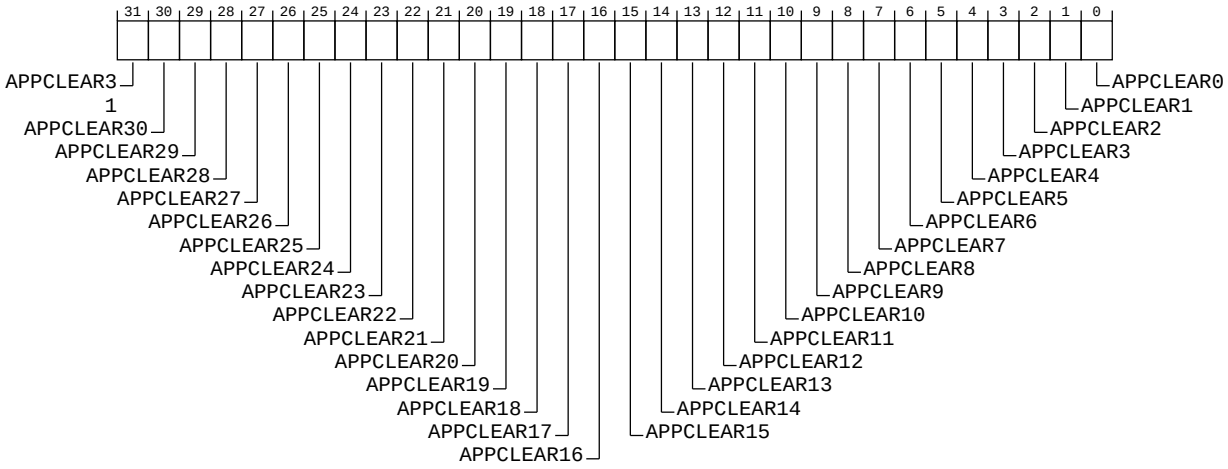


Table B-156: CTIAPPCLEAR bit descriptions

Bits	Name	Description	Reset
[31:0]	APPCLEAR<x>, bit[x], where x = 31 to 0	Application trigger <x> disable. Bits [31:N] are **RAZ/WI**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. Writing to this bit has the following effect: **0b0** No effect. **0b1** Clear corresponding application trigger to 0 and clear the corresponding channel event. If the ECT does not support multicycle channel events, use of CTIAPPCLEAR is deprecated and the debugger must only use ext-CTIAPPPULSE.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x018	CTIAPPCLEAR	None

This interface is accessible as follows:

When SoftwareLockStatus()
WI

When !SoftwareLockStatus()
WO

B.4.5 CTIAPPPULSE, CTI Application Pulse register

Causes event pulses to be generated on ECT channels.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
CTI

Register offset
0x01C

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-89: ext_ctiapppulse bit assignments

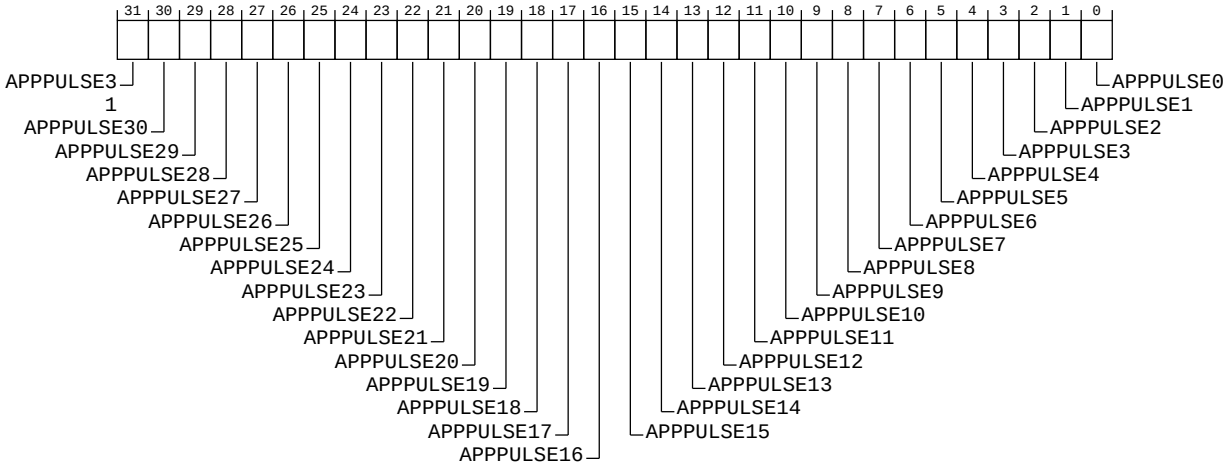


Table B-158: CTIAPPPULSE bit descriptions

Bits	Name	Description	Reset
[31:0]	APPPULSE<x>, bit[x], where x = 31 to 0	Generate event pulse on ECT channel <x>. Bits [31:N] are **RAZ/WI**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. Writing to this bit has the following effect: **0b0** No effect. **0b1** Channel <x> event pulse generated. • The CTIAPPPULSE operation does not affect the state of the application trigger. If the channel is active, either because of an earlier event or from the application trigger, then the value written to CTIAPPPULSE might have no effect. • Multiple pulse events that occur close together might be merged into a single pulse event.	32{x}

Accessibility

It is CONSTRAINED UNPREDICTABLE whether a write to CTIAPPPULSE generates an event on a channel if CTICONTROL.GLBEN is 0.

Component	Offset	Instance	Range
CTI	0x01C	CTIAPPPULSE	None

This interface is accessible as follows:

When SoftwareLockStatus()

WI

When !SoftwareLockStatus()

WO

B.4.6 CTIINEN<n>, CTI Input Trigger to Output Channel Enable registers, n = 0 - 31

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

Configurations

If input trigger n is not implemented or not connected, CTIINEN<n> is RES0.

Attributes

Width

32

Component

CTI

Register offset

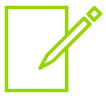
0x020 + (4 * n)

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-90: ext_ctiinen_n_ bit assignments

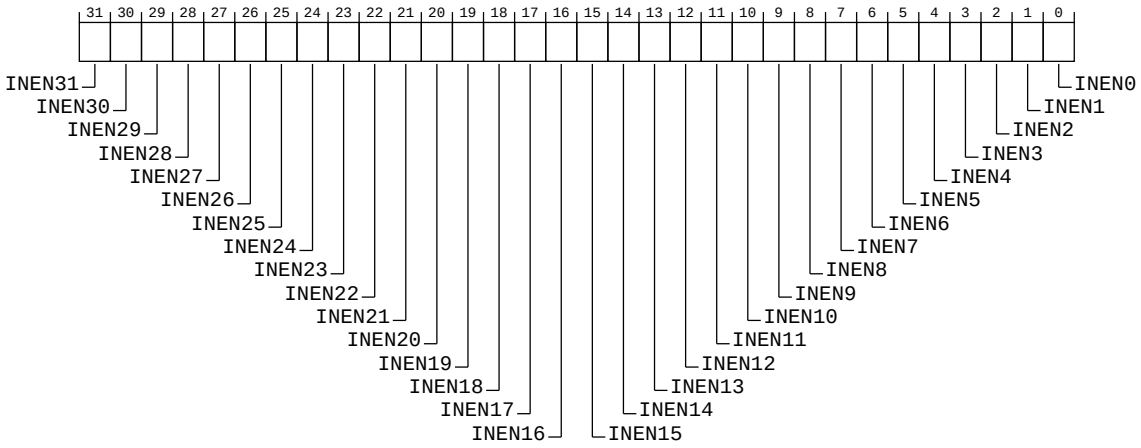


Table B-160: CTIINEN<n> bit descriptions

Bits	Name	Description	Reset
[31:0]	INEN<x>, bit[x], where x = 31 to 0	Input trigger <n> to output channel <x> enable. Bits [31:N] are **RAZ/WI**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. **0b0** Input trigger <n> will not generate an event on output channel <x>. **0b1** Input trigger <n> will generate an event on output channel <x>.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x020 + (4 * n)	CTIINEN<n>	None

This interface is accessible as follows:

When SoftwareLockStatus()

RO

When !SoftwareLockStatus()

RW

B.4.7 CTIOUTEN<n>, CTI Input Channel to Output Trigger Enable registers, n = 0 - 31

Defines which input channels generate output trigger n.

Configurations

If output trigger n is not implemented or not connected, CTIOUTEN<n> is RES0.

Attributes

Width

32

Component

CTI

Register offset

0x0A0 + (4 * n)

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-91: ext_ctiouten_n_ bit assignments

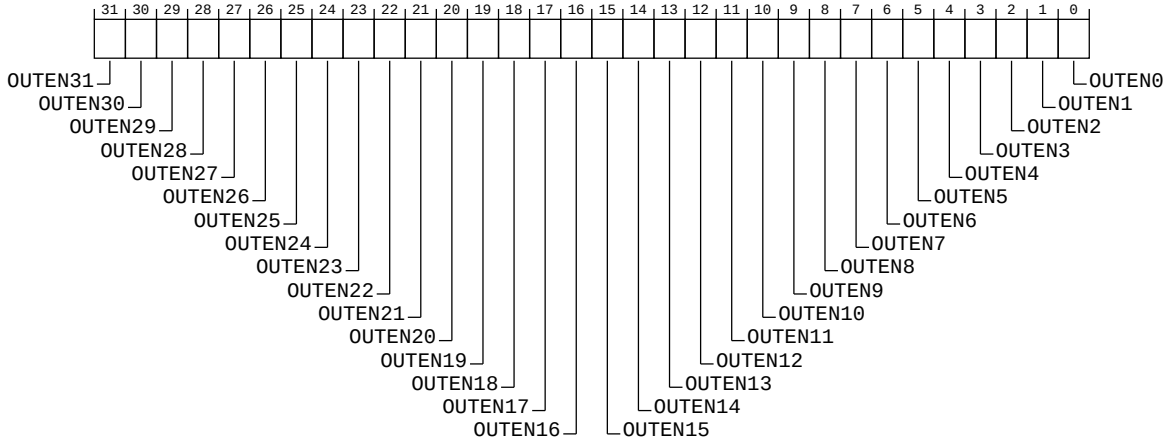


Table B-162: CTIOUTEN<n> bit descriptions

Bits	Name	Description	Reset
[31:0]	OUTEN<x>, bit[x], where x = 31 to 0	Input channel <x> to output trigger <n> enable. Bits [31:N] are **RAZ/WI**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. Possible values of this bit are: **0b0** An event on input channel <x> will not cause output trigger <n> to be asserted. **0b1** An event on input channel <x> will cause output trigger <n> to be asserted.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x0A0 + (4 * n)	CTIOUTEN<n>	None

This interface is accessible as follows:

When SoftwareLockStatus()
RO

When !SoftwareLockStatus()
RW

B.4.8 CTITRIGINSTATUS, CTI Trigger In Status register

Provides the status of the trigger inputs.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
CTI

Register offset
0x130

Access type
RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-92: ext_ctitriginstatus bit assignments

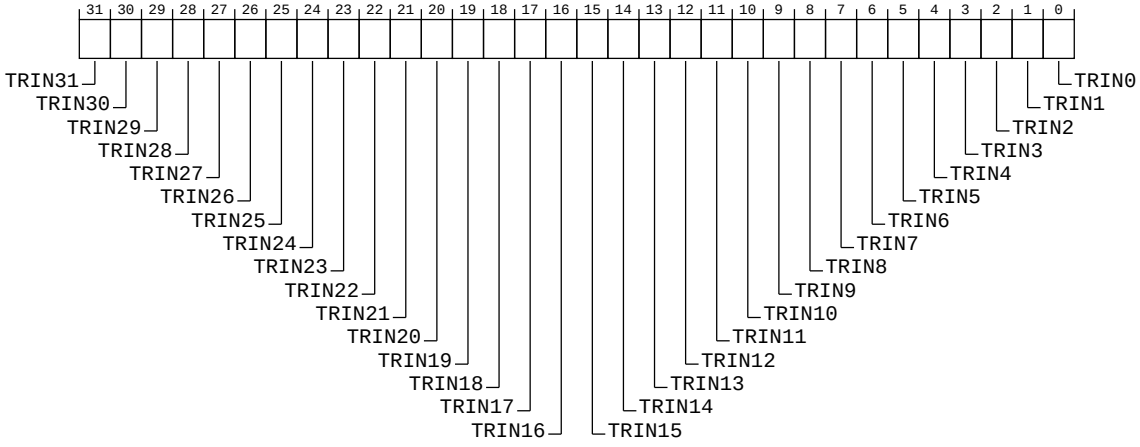


Table B-164: CTITRIGINSTATUS bit descriptions

Bits	Name	Description	Reset
[31:0]	TRIN<n>, bit[n], where n = 31 to 0	Trigger input <n> status. Bits [31:N] are **RAZ**. N is the number of CTI triggers implemented as defined by the ext-CTIDEVID.NUMTRIG field. **0b0** Input trigger n is inactive. **0b1** Input trigger n is active. Not implemented and not-connected input triggers are always inactive. It is IMPLEMENTATION DEFINED whether an input trigger that does not support multicycle events can be observed as active.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x130	CTITRIGINSTATUS	None

This interface is accessible as follows:

RO

B.4.9 CTITRIGOUTSTATUS, CTI Trigger Out Status register

Provides the raw status of the trigger outputs, after processing by any **IMPLEMENTATION DEFINED** trigger interface logic. For output triggers that are self-acknowledging, this is only meaningful if the CTI implements multicycle channel events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x134

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-93: ext_ctitrigoutstatus bit assignments

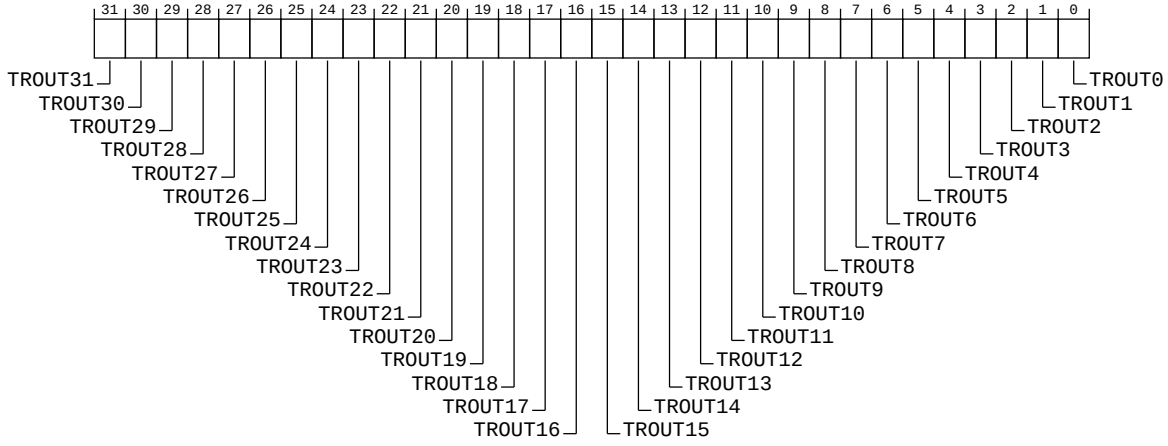


Table B-166: CTITRIGOUTSTATUS bit descriptions

Bits	Name	Description	Reset
[31:0]	TROUT<n>, bit[n], where n = 31 to 0	Trigger output <n> status. Bits [31:N] are **RAZ**. N is the value in ext-CTIDEVID.NUMTRIG. If n < N, and output trigger <n> is implemented and connected, and either the trigger is not self-acknowledging or the CTI implements multicycle channel events, then permitted values for TROUT<n> are: **0b0** Output trigger n is inactive. **0b1** Output trigger n is active. Otherwise when n < N it is **IMPLEMENTATION DEFINED** whether TROUT<n> behaves as described here or is RAZ.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x134	CTITRIGOUTSTATUS	None

This interface is accessible as follows:

RO

B.4.10 CTICHINSTATUS, CTI Channel In Status register

Provides the raw status of the ECT channel inputs to the CTI.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x138

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-94: ext_ctichinstatus bit assignments

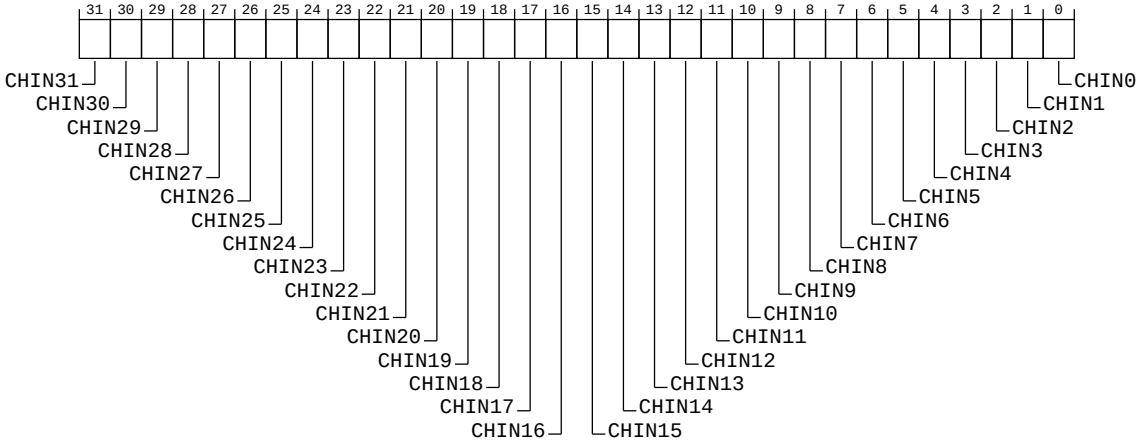


Table B-168: CTICHINSTATUS bit descriptions

Bits	Name	Description	Reset
[31:0]	CHIN<n>, bit[n], where n = 31 to 0	Input channel <n> status. Bits [31:N] are **RAZ**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. **0b0** Input channel <n> is inactive. **0b1** Input channel <n> is active. If the ECT channels do not support multicycle events then it is **IMPLEMENTATION DEFINED** whether an input channel can be observed as active.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x138	CTICHINSTATUS	None

This interface is accessible as follows:

RO

B.4.11 CTICHOUTSTATUS, CTI Channel Out Status register

Provides the status of the ECT channel outputs from the CTI.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x13C

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-95: ext_ctichoutstatus bit assignments

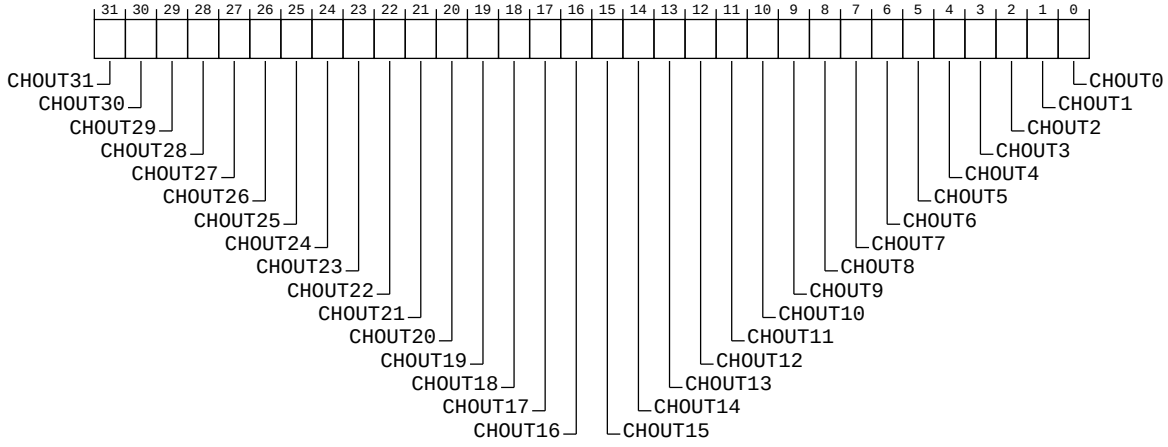


Table B-170: CTICHOUTSTATUS bit descriptions

Bits	Name	Description	Reset
[31:0]	CHOUT<n>, bit[n], where n = 31 to 0	Output channel <n> status. Bits [31:N] are **RAZ**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. Possible values of this bit are: **0b0** Output channel <n> is inactive. **0b1** Output channel <n> is active. If the ECT channels do not support multicycle events then it is **IMPLEMENTATION DEFINED** whether an output channel can be observed as active. **Note:** The value in CTICHOUTSTATUS is after gating by the channel gate. For more information, see ext-CTIGATE.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x13C	CTICHOUTSTATUS	None

This interface is accessible as follows:

RO

B.4.12 CTIGATE, CTI Channel Gate Enable register

Determines whether events on channels propagate through the CTM to other ECT components, or from the CTM into the CTI.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x140

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-96: ext_ctigate bit assignments

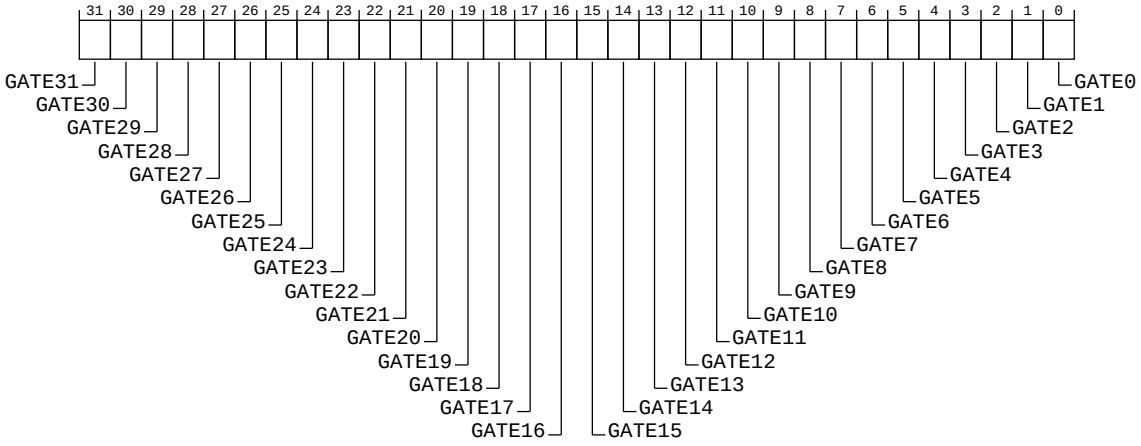


Table B-172: CTIGATE bit descriptions

Bits	Name	Description	Reset
[31:0]	GATE<x>, bit[x], where x = 31 to 0	Channel <x> gate enable. Bits [31:N] are **RAZ/WI**. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. **0b0** Disable output and, if ext-CTIDEVID.INOUT == 0b01, input channel <x> propagation. **0b1** Enable output and, if ext-CTIDEVID.INOUT == 0b01, input channel <x> propagation. If GATE<x> is set to 0, no new events will be propagated to the ECT, and if the ECT supports multicycle channel events any existing output channel events will be terminated.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x140	CTIGATE	None

This interface is accessible as follows:

When SoftwareLockStatus()

RO

When !SoftwareLockStatus()

RW

B.4.13 ASICCTL, CTI External Multiplexer Control register

Can be used to provide **IMPLEMENTATION DEFINED** controls for the CTI. For example, the register might be used to control multiplexors for additional **IMPLEMENTATION DEFINED** triggers. The **IMPLEMENTATION DEFINED** controls provided by this register might modify the architecturally defined behavior of the CTI.


Note

The architecturally-defined triggers must not be multiplexed.

Configurations

If it is implemented in the Core power domain then it is **IMPLEMENTATION DEFINED** whether it is in the Cold reset domain or the Warm reset domain.

This register must reset to a value that supports the architecturally-defined behavior of the CTI. Changing the value of the register from its reset value causes IMPLEMENTATION DEFINED behavior that might differ from the architecturally-defined behavior of the CTI.

Other than the requirements listed in this register description, all aspects of the reset behavior of the ASICCTL are IMPLEMENTATION DEFINED.

Attributes

Width

32

Component

CTI

Register offset

0x144

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-97: ext_asicctl bit assignments

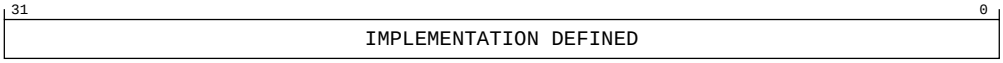


Table B-174: ASICCTL bit descriptions

Bits	Name	Description	Reset
[31:0]	None	None	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x144	ASICCTL	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

Otherwise

ImplementationDefined

B.4.14 CTIDEVCTL, CTI Device Control register

Provides target-specific device controls

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x150

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-98: ext_ctidevctl bit assignments

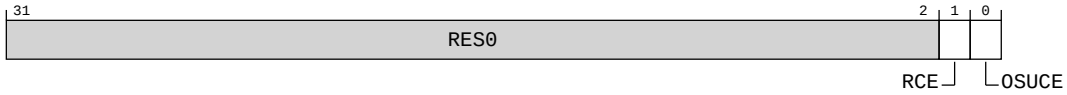


Table B-176: CTIDEVCTL bit descriptions

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**
[1]	RCE	Reset Catch Enable. **0b0** Reset Catch debug event disabled. **0b1** Reset Catch debug event enabled.	0b0
[0]	OSUCE	OS Unlock Catch Enable **0b0** OS Unlock Catch debug event disabled. **0b1** OS Unlock Catch debug event enabled.	0b0

Accessibility

Component	Offset	Instance	Range
CTI	0x150	CTIDEVCTL	None

This interface is accessible as follows:

When SoftwareLockStatus()

RO

When !SoftwareLockStatus()

RW

B.4.15 CTIDEVAFF0, CTI Device Affinity register 0

Copy of the low half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

Configurations

If the CTI is CTIv1, this register is OPTIONAL. If the CTI is CTIv2, this register is mandatory.

Arm recommends that the CTI is CTIv2.

In an Armv8.5 compliant implementation, the CTI must be CTIv2.

If this register is implemented, then ext-CTIDEVAFF1 must also be implemented. If the CTI of a PE does not implement the CTI Device Affinity registers, the CTI block of the external debug memory map must be located 64KB above the debug registers in the external debug interface.

Attributes

Width

32

Component

CTI

Register offset

0xFA8

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-99: ext_ctidevaff0 bit assignments

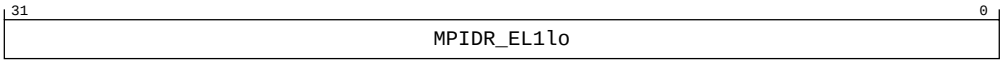


Table B-178: CTIDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1lo	AArch64-MPIDR_EL1 low half. Read-only copy of the low half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0xFA8	CTIDEVAFF0	None

This interface is accessible as follows:

RO

B.4.16 CTIDEVAFF1, CTI Device Affinity register 1

Copy of the high half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

Configurations

If the CTI is CTIv1, this register is OPTIONAL. If the CTI is CTIv2, this register is mandatory.

Arm recommends that the CTI is CTIv2.

In an Armv8.5 compliant implementation, the CTI must be CTIv2.

If this register is implemented, then ext-CTIDEVAFF0 must also be implemented. If the CTI of a PE does not implement the CTI Device Affinity registers, the CTI block of the external debug memory map must be located 64KB above the debug registers in the external debug interface.

Attributes

Width

32

Component

CTI

Register offset

0xFAC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-100: ext_ctidevaff1 bit assignments

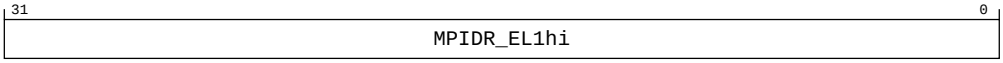


Table B-180: CTIDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1hi	AArch64-MPIDR_EL1 high half. Read-only copy of the high half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32{x}

Accessibility

Component	Offset	Instance	Range
CTI	0xFAC	CTIDEVAFF1	None

This interface is accessible as follows:

RO

B.4.17 CTILAR, CTI Lock Access Register

Allows or disallows access to the CTI registers through a memory-mapped interface.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Cross-Trigger Interface registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Cross-Trigger Interface registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

If FEAT_Debugv8p4 is implemented, the Software Lock is not implemented.

Software uses CTILAR to set or clear the lock, and ext-CTILSR to check the current status of the lock.

Attributes

Width

32

Component

CTI

Register offset

0xFB0

Access type

RESERVEDW

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-101: ext_ctilar bit assignments

Table B-182: CTILAR bit descriptions

Bits	Name	Description	Reset
[31:0]	KEY	Lock Access control. Writing the key value `0xC5ACCE55` to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.	`32{x}`

Accessibility

Component	Offset	Instance	Range
CTI	0xFB0	CTILAR	None

This interface is accessible as follows:

WO

B.4.18 CTILSR, CTI Lock Status Register

Indicates the current status of the Software Lock for CTI registers.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Cross-Trigger Interface registers. Use of this lock mechanism reduces the risk of accidental damage to the

contents of the Cross-Trigger Interface registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

If FEAT_Debugv8p4 is implemented, the Software Lock is not implemented.

Software uses ext-CTILAR to set or clear the lock, and CTILSR to check the current status of the lock.

Attributes

Width

32

Component

CTI

Register offset

0xFB4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx1x

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-102: ext_ctilsr bit assignments

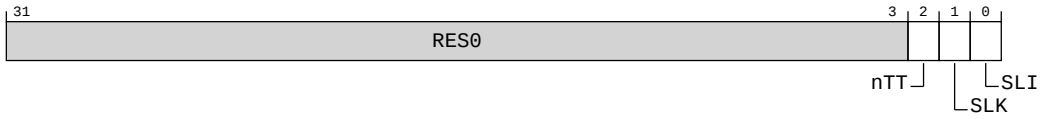


Table B-184: CTILSR bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2]	nTT	Not thirty-two bit access required. **RAZ**.	x

Bits	Name	Description	Reset
[1]	SLK	Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is **RES0**. For memory-mapped accesses when the Software Lock is implemented, possible values of this field are: **0b0** Lock clear. Writes are permitted to this component's registers. **0b1** Lock set. Writes to this component's registers are ignored, and reads have no side effects.	0b1
[0]	SLI	Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is **IMPLEMENTATION DEFINED**. Permitted values are: **0b0** Software Lock not implemented or not memory-mapped access. **0b1** Software Lock implemented and memory-mapped access.	x

Accessibility

Component	Offset	Instance	Range
CTI	0xFB4	CTILSR	None

This interface is accessible as follows:

RO

B.4.19 CTIAUTHSTATUS, CTI Authentication Status register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for CTI.

Configurations

This register is OPTIONAL, and is required for CoreSight compliance.

Attributes

Width

32

Component

CTI

Register offset

0xFB8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-103: ext_ctiauthstatus bit assignments

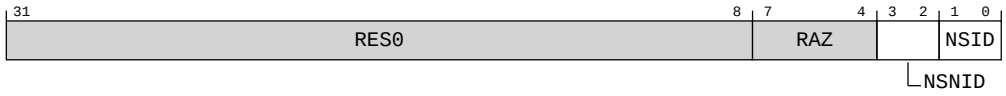


Table B-186: CTIAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	**RAZ**	Reserved	**RAZ**
[3:2]	NSNID	If EL3 is implemented, this field holds the same value as ext-DBGAUTHSTATUS_EL1.NSNID.	xx
[1:0]	NSID	If EL3 is implemented, this field holds the same value as ext-DBGAUTHSTATUS_EL1.NSID.	xx

Accessibility

Component	Offset	Instance	Range
CTI	0xFB8	CTIAUTHSTATUS	None

This interface is accessible as follows:

RO

B.4.20 CTIDEVARCH, CTI Device Architecture register

Identifies the programmers' model architecture of the CTI component.

Configurations

If the CTI is CTIv1, this register is OPTIONAL. If the CTI is CTIv2, this register is mandatory.

Arm recommends that the CTI is CTIv2.

In an Armv8.5 compliant implementation, the CTI must be CTIv2.

If this register is not implemented, ext-CTIDEVAFF0 and ext-CTIDEVAFF1 are also not implemented.

Attributes

Width

32

Component

CTI

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 xxxx 0001 1010 0001 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-104: ext_ctidevarch bit assignments

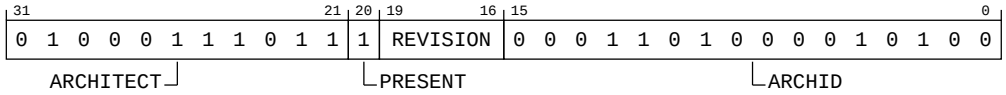


Table B-188: CTIDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For CTI, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0x4`. Bits [27:21] are the JEP106 ID code, `0x3B`. **`0b01000111011`**	`0b01000111011`
[20]	PRESENT	Indicates that the DEVARCH is present. **`0b1`**	`0b1`

Bits	Name	Description	Reset
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0b0000** First revision. **0b0001** As 0b0000, and also adds support for ext-CTIDEVCTL. All other values are reserved.	The reset values can be the following: `0b0000`, `0b0001`, respective to the value.
[15:0]	ARCHID	Defines this part to be an Armv8 debug component. For architectures defined by Arm this is further subdivided. For CTI: • Bits [15:12] are the architecture version, `0x1`. • Bits [11:0] are the architecture part number, `0xA14`. This corresponds to CTI architecture version CTIv2. **0b0001101000010100**	`0x1A14`

Accessibility

Component	Offset	Instance	Range
CTI	0xFBC	CTIDEVARCH	None

This interface is accessible as follows:

RO

B.4.21 CTIDEVID2, CTI Device ID register 2

Reserved for future information about the CTI component to the debugger.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFC0

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-105: ext_ctidevid2 bit assignments

Table B-190: CTIDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
CTI	0xFC0	CTIDEVID2	None

This interface is accessible as follows:

RO

B.4.22 CTIDEVID1, CTI Device ID register 1

Reserved for future information about the CTI component to the debugger.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFC4

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-106: ext_ctidevid1 bit assignments

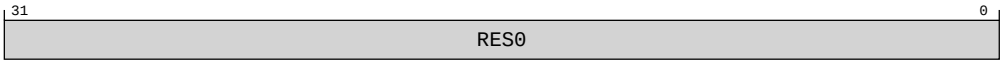


Table B-192: CTIDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
CTI	0xFC4	CTIDEVID1	None

This interface is accessible as follows:

RO

B.4.23 CTIDEVID, CTI Device ID register 0

Describes the CTI component to the debugger.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFC8

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-107: ext_ctidevid bit assignments

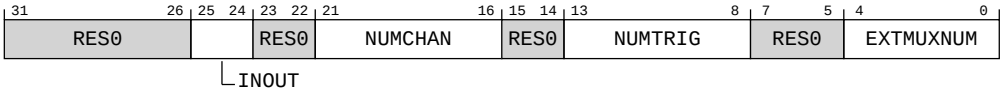


Table B-194: CTIDEVID bit descriptions

Bits	Name	Description	Reset
[31:26]	**RES0**	Reserved	**RES0**
[25:24]	INOUT	Input/output options. Indicates presence of the input gate. If the CTM is not implemented or CTIv2 is not implemented, this field is **RAZ**. **0b00** ext-CTIGATE does not mask propagation of input events from external channels. **0b01** ext-CTIGATE masks propagation of input events from external channels. All other values are reserved.	xx
[23:22]	**RES0**	Reserved	**RES0**
[21:16]	NUMCHAN	Number of ECT channels implemented. For Armv8, valid values are: • 0b000011 3 channels (0..2) implemented. • 0b000100 4 channels (0..3) implemented. • 0b000101 5 channels (0..4) implemented. • 0b000110 6 channels (0..5) implemented. and so on up to 0b100000, 32 channels (0..31) implemented. All other values are reserved.	6{x}
[15:14]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[13:8]	NUMTRIG	Upper bound for number of triggers. The indices of all implemented input and output triggers are less than this value. All other values are reserved. If the PE contains a Trace extension, this field must be at least `0b001000`. There is no guarantee that all of the input and output triggers, including the highest numbered, are connected to any components, or that the implementation of input and output triggers is symmetrical.	`6{x}`
[7:5]	**RES0**	Reserved	**RES0**
[4:0]	EXTMUXNUM	Number of multiplexors available on triggers. This value is used in conjunction with External Control register, ext-ASICCTL.	`5{x}`

Accessibility

Component	Offset	Instance	Range
CTI	0xFC8	CTIDEVID	None

This interface is accessible as follows:

RO

B.5 External Debug registers summary

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

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

Table B-196: Debug registers summary

Offset	Name	Reset	Width	Description
0x020	EDESR	–	32-bit	External Debug Event Status Register
0x024	EDECR	–	32-bit	External Debug Execution Control Register
0x030	EDWAR [31:0]	–	32-bit	External Debug Watchpoint Address Register
0x034	EDWAR [63:32]	–	32-bit	External Debug Watchpoint Address Register
0x080	DBGDTRRX_EL0	–	32-bit	Debug Data Transfer Register, Receive
0x084	EDITR	–	32-bit	External Debug Instruction Transfer Register
0x088	EDSCR	–	32-bit	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	–	32-bit	Debug Data Transfer Register, Transmit
0x090	EDRCR	–	32-bit	External Debug Reserve Control Register
0x098	EDECCR	–	32-bit	External Debug Exception Catch Control Register
0x300	OSLAR_EL1	–	32-bit	OS Lock Access Register
0x310	EDPRCR	–	32-bit	External Debug Power/Reset Control Register
0x314	EDPRSR	–	32-bit	External Debug Processor Status Register
0x400	DBGBVR0_EL1 [63:0]	–	64-bit	Debug Breakpoint Value Registers
0x408	DBGBCR0_EL1	–	32-bit	Debug Breakpoint Control Registers

Offset	Name	Reset	Width	Description
0x410	DBGBVR1_EL1 [63:0]	—	64-bit	Debug Breakpoint Value Registers
0x418	DBGBCR1_EL1	—	32-bit	Debug Breakpoint Control Registers
0x420	DBGBVR2_EL1 [63:0]	—	64-bit	Debug Breakpoint Value Registers
0x428	DBGBCR2_EL1	—	32-bit	Debug Breakpoint Control Registers
0x430	DBGBVR3_EL1 [63:0]	—	64-bit	Debug Breakpoint Value Registers
0x438	DBGBCR3_EL1	—	32-bit	Debug Breakpoint Control Registers
0x440	DBGBVR4_EL1 [63:0]	—	64-bit	Debug Breakpoint Value Registers
0x448	DBGBCR4_EL1	—	32-bit	Debug Breakpoint Control Registers
0x450	DBGBVR5_EL1 [63:0]	—	64-bit	Debug Breakpoint Value Registers
0x458	DBGBCR5_EL1	—	32-bit	Debug Breakpoint Control Registers
0x800	DBGWVR0_EL1 [63:0]	—	64-bit	Debug Watchpoint Value Registers
0x808	DBGWCR0_EL1	—	32-bit	Debug Watchpoint Control Registers
0x810	DBGWVR1_EL1 [63:0]	—	64-bit	Debug Watchpoint Value Registers
0x818	DBGWCR1_EL1	—	32-bit	Debug Watchpoint Control Registers
0x820	DBGWVR2_EL1 [63:0]	—	64-bit	Debug Watchpoint Value Registers
0x828	DBGWCR2_EL1	—	32-bit	Debug Watchpoint Control Registers
0x830	DBGWVR3_EL1 [63:0]	—	64-bit	Debug Watchpoint Value Registers
0x838	DBGWCR3_EL1	—	32-bit	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	—	32-bit	Main ID Register
0xD20	EDPFR [31:0]	—	32-bit	External Debug Processor Feature Register
0xD24	EDPFR [63:32]	—	32-bit	External Debug Processor Feature Register
0xD28	EDDFR [31:0]	—	32-bit	External Debug Feature Register
0xD2C	EDDFR [63:32]	—	32-bit	External Debug Feature Register
0xD60	EDAA32PFR	—	64-bit	External Debug Auxiliary Processor Feature Register
0xF00	EDITCTRL	—	32-bit	External Debug Integration mode Control register
0xFA0	DBGCLAIMSET_EL1	—	32-bit	Debug CLAIM Tag Set register
0xFA4	DBGCLAIMCLR_EL1	—	32-bit	Debug CLAIM Tag Clear register
0xFA8	EDDEVAFF0	—	32-bit	External Debug Device Affinity register 0
0xFAC	EDDEVAFF1	—	32-bit	External Debug Device Affinity register 1
0xFB0	EDLAR	—	32-bit	External Debug Lock Access Register
0xFB4	EDLSR	—	32-bit	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	—	32-bit	Debug Authentication Status register
0xFBC	EDDEVARCH	—	32-bit	External Debug Device Architecture register
0xFC0	EDDEVID2	—	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	—	32-bit	External Debug Device ID register 1
0xFC8	EDDEVID	—	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	—	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	—	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	—	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	—	32-bit	External Debug Peripheral Identification Register 1

Offset	Name	Reset	Width	Description
0xFE8	EDPIDR2	–	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	–	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	–	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	–	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	–	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	–	32-bit	External Debug Component Identification Register 3

B.5.1 EDESR, External Debug Event Status Register

Indicates the status of internally pending Halting debug events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x020

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-108: ext_edesr bit assignments

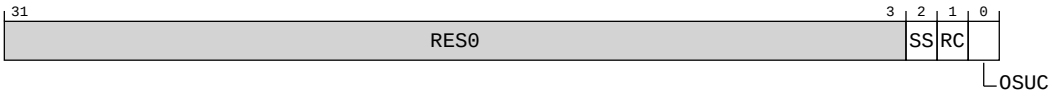


Table B-197: EDESR bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2]	SS	Halting step debug event pending. Possible values of this field are: **0b0** Reading this means that a Halting step debug event is not pending. Writing this means no action. **0b1** Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.	0b0
[1]	RC	Reset Catch debug event pending. Possible values of this field are: **0b0** Reading this means that a Reset Catch debug event is not pending. Writing this means no action. **0b1** Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.	0b0
[0]	OSUC	OS Unlock Catch debug event pending. Possible values of this field are: **0b0** Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action. **0b1** Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.	0b0

Access

If a request to clear a pending Halting debug event is received at or about the time when halting becomes allowed, it is **CONSTRAINED UNPREDICTABLE** whether the event is taken.

If Core power is removed while a Halting debug event is pending, it is lost. However, it might become pending again when the Core is powered back on and Cold reset.

Accessibility

If a request to clear a pending Halting debug event is received at or about the time when halting becomes allowed, it is CONSTRAINED UNPREDICTABLE whether the event is taken.

If Core power is removed while a Halting debug event is pending, it is lost. However, it might become pending again when the Core is powered back on and Cold reset.

Component	Offset	Instance	Range
Debug	0x020	EDESR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !SoftwareLockStatus()

RW

Otherwise
ERROR

B.5.2 EDECR, External Debug Execution Control Register

Controls Halting debug events.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width
32

Component
Debug

Register offset
0x024

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-109: ext_edecr bit assignments

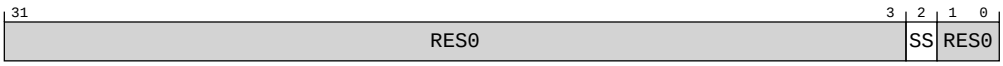


Table B-199: EDECR bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[2]	SS	Halting step enable. Possible values of this field are: **0b0** Halting step debug event disabled. **0b1** Halting step debug event enabled. If the value of EDECR.SS is changed when the PE is in Non-debug state, behavior is **CONSTRAINED UNPREDICTABLE** as described in *Changing the value of EDECR.SS when not in Debug state* in the *Arm® Architecture Reference Manual for A-profile architecture*.	0b0
[1:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0x024	EDECR	None

This interface is accessible as follows:

When IsCorePowered() && SoftwareLockStatus()
RO

When IsCorePowered() && !SoftwareLockStatus()
RW

Otherwise
ERROR

B.5.3 EDWAR, External Debug Watchpoint Address Register

Returns the virtual data address being accessed when a Watchpoint Debug Event was triggered.

Configurations

This register is available in all configurations.

Attributes

Width
64

Component
Debug

Register offsets (2)
0x030,0x034

Access type
See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-110: ext_edwar bit assignments

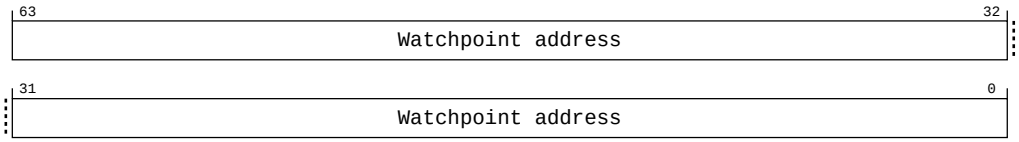


Table B-201: EDWAR bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Watchpoint address. The data virtual address being accessed when a Watchpoint Debug Event was triggered and caused entry to Debug state. This address must be within a naturally-aligned block of memory of power-of-two size no larger than the DC ZVA block size. The value of this register is **UNKNOWN** if the PE is in Non-debug state, or if Debug state was entered other than for a Watchpoint debug event. The EDWAR is subject to the same alignment rules as the reporting of a watchpointed address in the FAR. See *Determining the memory location that caused a Watchpoint exception* in the *Arm® Architecture Reference Manual for A-profile architecture*.	64{x}

Accessibility

Component	Offset	Instance	Range
Debug	0x030	EDWAR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
Debug	0x034	EDWAR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus()

RO

Otherwise

ERROR

B.5.4 DBGDTRRX_EL0, Debug Data Transfer Register, Receive

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. See AArch64-DBGDTR_EL0 for additional architectural mappings. It is a component of the Debug Communications Channel.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x080

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-111: ext_dbgdtrrx_el0 bit assignments

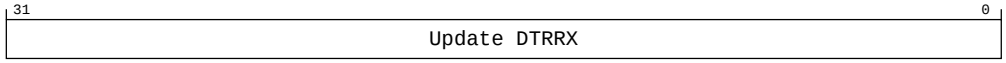


Table B-204: DBGDTRRX_EL0 bit descriptions

Bits	Name	Description	Reset
[31:0]	None	Update DTRRX. Writes to this register: • If RXfull is set to 1, set DTRRX to **UNKNOWN**. • If RXfull is set to 0, update the value in DTRRX. After the write, RXfull is set to 1. Reads of this register: • If RXfull is set to 1, return the last value written to DTRRX. • If RXfull is set to 0, return an **UNKNOWN** value. After the read, RXfull remains unchanged. For the full behavior of the Debug Communications Channel, see *The Debug Communication Channel and Instruction Transfer Register* in the *Arm® Architecture Reference Manual for A-profile architecture*.	32{x}

Access

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

Accessibility

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is CONSTRAINED UNPREDICTABLE, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an UNKNOWN state.

Component	Offset	Instance	Range
Debug	0x080	DBGDTRRX_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.5 EDITR, External Debug Instruction Transfer Register

Used in Debug state for passing instructions to the PE for execution.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x084

Access type

See bit descriptions

Reset value

When in AArch64 state

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

When in AArch64 state

Figure B-112: ext_editr bit assignments

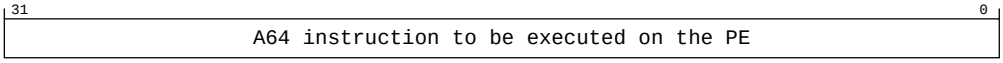


Table B-206: EDITR bit descriptions

Bits	Name	Description	Reset
[31:0]	None	A64 instruction to be executed on the PE.	32{x}

Access

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any instruction issued through the ITR in Normal access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

EDITR ignores writes if the PE is in Non-debug state.

Accessibility

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any instruction issued through the ITR in Normal access mode that has not completed execution is CONSTRAINED UNPREDICTABLE, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an UNKNOWN state.

EDITR ignores writes if the PE is in Non-debug state.

Component	Offset	Instance	Range
Debug	0x084	EDITR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()
WI

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()
WO

Otherwise
ERROR

B.5.6 EDSCR, External Debug Status and Control Register

Main control register for the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x088

Access type

See bit descriptions

Reset value

000x 00xx 0000 xxxx x0xx xxxx x0xx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-113: ext_edscr bit assignments

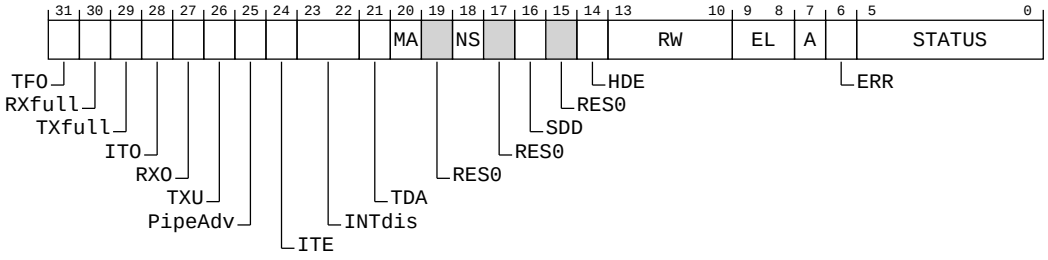


Table B-208: EDSCR bit descriptions

Bits	Name	Description	Reset
[31]	TFO	Trace Filter Override. Overrides the Trace Filter controls allowing the external debugger to trace any visible Exception level. **0b0** Trace Filter controls are not affected. **0b1** Trace Filter controls in AArch64-TRFCR_EL1 and AArch64-TRFCR_EL2 are ignored. When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers: • AArch64-MDSCR_EL1. This bit is ignored by the PE when ExternalSecureNoninvasiveDebugEnabled() is FALSE and the Effective value of AArch64-MDCR_EL3.STE is 1.	0b0
[30]	RXfull	DTRRX full. Access to this field is: RO	0b0
[29]	TXfull	DTRTX full. Access to this field is: RO	0b0
[28]	ITO	ITR overrun. Set to 0 on entry to Debug state. **When !Halted()** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x
[27]	RXO	DTRRX overrun. Access to this field is: RO	0b0
[26]	TXU	DTRTX underrun. Access to this field is: RO	0b0
[25]	PipeAdv	Pipeline Advance. Indicates that software execution is progressing. **0b0** No progress has been made by the PE since the last time this field was cleared to zero by writing 1 to ext-EDRCR.CSPA. **0b1** Progress has been made by the PE since the last time this field was cleared to zero by writing 1 to ext-EDRCR.CSPA. The architecture does not define precisely when this field is set to 1. It requires only that this happen periodically in Non-debug state to indicate that software execution is progressing. For example, a PE might set this field to 1 each time the PE retires one or more instructions, or at periodic intervals during the progression of an instruction. Access to this field is: RO	x

Bits	Name	Description	Reset
[24]	ITE	ITR empty. **When !Halted()** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x
[23:22]	INTdis	Interrupt disable. Disables taking interrupts in Non-debug state. **0b00** Masking of interrupts is controlled by PSTATE and interrupt routing controls. **0b01** If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked. If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked. **Note:** All interrupts includes virtual and SError interrupts. When AArch64-OSLSR_EL1.OSLK is 1, this field can be indirectly read and written through the following System registers: • AArch64-MDSCR_EL1. The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE. When FEAT_Debugv8p4 is implemented, bit[23] of this register is **RES0**.	0b00
[21]	TDA	Traps accesses to the following debug System registers: • AArch64: AArch64-DBGBCR<n>_EL1, AArch64-DBGBVR<n>_EL1, AArch64-DBGWCR<n>_EL1, AArch64-DBGWVR<n>_EL1. **0b0** Accesses to debug System registers do not generate a Software Access Debug event. **0b1** Accesses to debug System registers generate a Software Access Debug event, if AArch64-OSLSR_EL1.OSLK is 0 and if halting is allowed.	0b0
[20]	MA	Memory access mode. Controls the use of memory-access mode for accessing ITR and the DCC. This bit is ignored if in Non-debug state and set to zero on entry to Debug state. Possible values of this field are: **0b0** Normal access mode. **0b1** Memory access mode.	0b0
[19]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[18]	NS	Non-secure status. In Debug state, gives the current Security state: **0b0** Secure state. **0b1** Non-secure state. **When !Halted()** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x
[17]	**RES0**	Reserved	**RES0**
[16]	SDD	Secure debug disabled. On entry to Debug state: • If entering in Secure state, SDD is set to 0. • If entering in Non-secure state, SDD is set to the inverse of `ExternalSecureInvasiveDebugEnabled`(). In Debug state, the value of the SDD bit does not change, even if `ExternalSecureInvasiveDebugEnabled`() changes. In Non-debug state: • SDD returns the inverse of `ExternalSecureInvasiveDebugEnabled`(). If the authentication signals that control `ExternalSecureInvasiveDebugEnabled`() change, a context synchronization event is required to guarantee their effect. • This bit is unaffected by the Security state of the PE. Access to this field is: RO	x
[15]	**RES0**	Reserved	**RES0**
[14]	HDE	Halting debug enable. **0b0** Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events. **0b1** Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.	0b0
[13:10]	RW	Exception level Execution state status. In Debug state, each bit gives the current Execution state of each Exception level. **0b1111** Any of the following: • The PE is in Non-debug state. • The PE is at EL0 using AArch64. • The PE is not at EL0, and EL1, EL2, and EL3 are using AArch64. **When !Halted()** Access to this field is: **RAO/WI** **Otherwise** Access to this field is: RO	xxxx

Bits	Name	Description	Reset
[9:8]	EL	Exception level. In Debug state, gives the current Exception level of the PE. **When !Halted()** Access to this field is: **RAZ/WI** **Otherwise** Access to this field is: RO	xx
[7]	A	SError interrupt pending. In Debug state, indicates whether an SError interrupt is pending: • If AArch64-HCR_EL2.{AMO, TGE} = {1, 0}, EL2 is enabled in the current Security state, and the PE is executing at EL0 or EL1, a virtual SError interrupt. • Otherwise, a physical SError interrupt. **0b0** No SError interrupt pending. **0b1** SError interrupt pending. A debugger can read EDSCR to check whether an SError interrupt is pending without having to execute further instructions. A pending SError might indicate data from target memory is corrupted. **When !Halted()** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x
[6]	ERR	Cumulative error flag. This bit is set to 1 following exceptions in Debug state and on any signaled overrun or underrun on the DTR or EDITR. Access to this field is: RO	0b0

Bits	Name	Description	Reset
[5:0]	STATUS	Debug status flags. **0b000001** PE is restarting, exiting Debug state. **0b000010** PE is in Non-debug state. **0b000111** Breakpoint. **0b010011** External debug request. **0b011011** Halting step, normal. **0b011111** Halting step, exclusive. **0b100011** OS Unlock Catch. **0b100111** Reset Catch. **0b101011** Watchpoint. **0b101111** HLT instruction. **0b110011** Software access to debug register. **0b110111** Exception Catch. **0b111011** Halting step, no syndrome. All other values of STATUS are reserved. Access to this field is: RO	6{x}

Accessibility

Component	Offset	Instance	Range
Debug	0x088	EDSCR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()
RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()
RW

Otherwise

ERROR

B.5.7 DBGDTRTX_EL0, Debug Data Transfer Register, Transmit

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. See AArch64-DBGDTR_EL0 for additional architectural mappings. It is a component of the Debug Communication Channel.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x08C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-114: ext_dbgdtrtx_el0 bit assignments

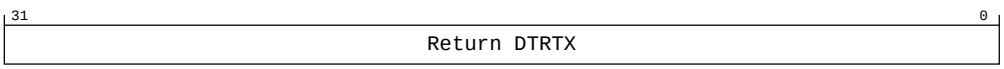


Table B-210: DBGDTRTX_EL0 bit descriptions

Bits	Name	Description	Reset
[31:0]	None	Return DTRTX. Reads of this register: • If TXfull is set to 1, return the last value written to DTRTX. • If TXfull is set to 0, return an **UNKNOWN** value. After the read, TXfull is cleared to 0. Writes to this register: • If TXfull is set to 1, set DTRTX to **UNKNOWN**. • If TXfull is set to 0, update the value in DTRTX. After the write, TXfull remains unchanged. For the full behavior of the Debug Communications Channel, see *The Debug Communication Channel and Instruction Transfer Register* in the *Arm® Architecture Reference Manual for A-profile architecture*.	32{x}

Access

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

Accessibility

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is CONSTRAINED UNPREDICTABLE, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an UNKNOWN state.

Component	Offset	Instance	Range
Debug	0x08C	DBGDTRTX_EL0	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.8 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in ext-EDSCR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x090

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-115: ext_edrcr bit assignments

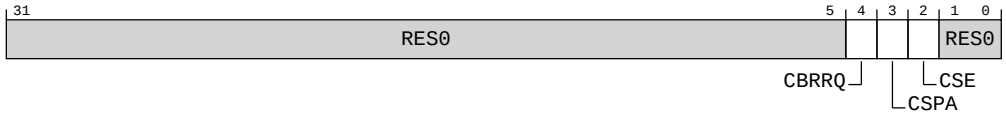


Table B-212: EDRCR bit descriptions

Bits	Name	Description	Reset
[31:5]	**RES0**	Reserved	**RES0**
[4]	CBRRQ	This feature is not supported. Writes to this bit are ignored **0b0** No action.	x

Bits	Name	Description	Reset
[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.	x
[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.	x
[1:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0x090	EDRCR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()
WI

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()
WO

Otherwise
ERROR

B.5.9 EDECCR, External Debug Exception Catch Control Register

Controls Exception Catch debug events. For more information, see *Exception Catch debug event* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
Debug

Register offset
0x098

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx x000 0000 x00x 000x

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-116: ext_edeccr bit assignments

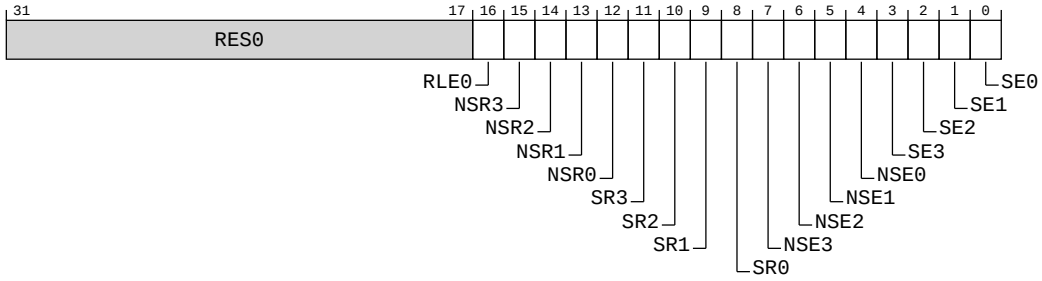


Table B-214: EDECCR bit descriptions

Bits	Name	Description	Reset
[31:17]	**RES0**	Reserved	**RES0**
[16]	RLE0	None Access to this field is: **RES0**	x
[15]	NSR3	None Access to this field is: **RES0**	x
[14]	NSR2	Controls exception catch on exception return to Non-secure EL2 in conjunction with EDECCR.NSE2. **0b0** If EDECCR.NSE2 is 0, then Exception Catch debug events are disabled for Non-secure EL2. If EDECCR.NSE2 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL2. **0b1** If EDECCR.NSE2 is 0, then Exception Catch debug events are enabled for exception returns to Non-secure EL2. If EDECCR.NSE2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL2.	0b0

Bits	Name	Description	Reset
[13]	NSR1	Controls exception catch on exception return to Non-secure EL1 in conjunction with EDECCR.NSE1. **0b0** If EDECCR.NSE1 is 0, then Exception Catch debug events are disabled for Non-secure EL1. If EDECCR.NSE1 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL1. **0b1** If EDECCR.NSE1 is 0, then Exception Catch debug events are enabled for exception returns to Non-secure EL1. If EDECCR.NSE1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL1.	0b0
[12]	NSR0	Controls exception catch on exception return to Non-secure EL0. **0b0** Exception Catch debug events are disabled for Non-secure EL0. **0b1** Exception Catch debug events are enabled for exception returns to Non-secure EL0.	0b0
[11]	SR3	Controls exception catch on exception return to EL3 in conjunction with EDECCR.SE3. **0b0** If EDECCR.SE3 is 0, then Exception Catch debug events are disabled for EL3. If EDECCR.SE3 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to EL3. **0b1** If EDECCR.SE3 is 0, then Exception Catch debug events are enabled for exception returns to EL3. If EDECCR.SE3 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to EL3.	0b0
[10]	SR2	Controls exception catch on exception return to Secure EL2 in conjunction with EDECCR.SE2. **0b0** If EDECCR.SE2 is 0, then Exception Catch debug events are disabled for Secure EL2. If EDECCR.SE2 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL2. **0b1** If EDECCR.SE2 is 0, then Exception Catch debug events are enabled for exception returns to Secure EL2. If EDECCR.SE2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL2.	0b0

Bits	Name	Description	Reset
[9]	SR1	Controls exception catch on exception return to Secure EL1 in conjunction with EDECCR.SE1. **0b0** If EDECCR.SE1 is 0, then Exception Catch debug events are disabled for Secure EL1. If EDECCR.SE1 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL1. **0b1** If EDECCR.SE1 is 0, then Exception Catch debug events are enabled for exception returns to Secure EL1. If EDECCR.SE1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL1.	0b0
[8]	SR0	Controls exception catch on exception return to Secure EL0. **0b0** Exception Catch debug events are disabled for Secure EL0. **0b1** Exception Catch debug events are enabled for exception returns to Secure EL0.	0b0
[7]	NSE3	None Access to this field is: **RES0**	x
[6]	NSE2	Controls exception catch on exception entry to Non-secure EL2. Also controls exception catch on exception return to Non-secure EL2 in conjunction with EDECCR.NSR2. **0b0** If EDECCR.NSR2 is 0, then Exception Catch debug events are disabled for Non-secure EL2. If EDECCR.NSR2 is 1, then Exception Catch debug events are enabled for exception returns to Non-secure EL2. **0b1** If EDECCR.NSR2 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL2. If EDECCR.NSR2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL2. **Note:** It is **IMPLEMENTATION DEFINED** whether a reset entry to an Exception level will generate an Exception Catch debug event.	0b0

Bits	Name	Description	Reset
[5]	NSE1	Controls exception catch on exception entry to Non-secure EL1. Also controls exception catch on exception return to Non-secure EL1 in conjunction with EDECCR.NSR1. **0b0** If EDECCR.NSR1 is 0, then Exception Catch debug events are disabled for Non-secure EL1. If EDECCR.NSR1 is 1, then Exception Catch debug events are enabled for exception returns to Non-secure EL1. **0b1** If EDECCR.NSR1 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Non-secure EL1. If EDECCR.NSR1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure EL1. **Note:** It is **IMPLEMENTATION DEFINED** whether a reset entry to an Exception level will generate an Exception Catch debug event.	0b0
[4]	NSE0	None Access to this field is: **RES0**	x
[3]	SE3	Controls exception catch on exception entry to EL3. Also controls exception catch on exception return to EL3 in conjunction with EDECCR.SR3. **0b0** If EDECCR.SR3 is 0, then Exception Catch debug events are disabled for EL3. If EDECCR.SR3 is 1, then Exception Catch debug events are enabled for exception returns to EL3. **0b1** If EDECCR.SR3 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to EL3. If EDECCR.SR3 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to EL3. **Note:** It is **IMPLEMENTATION DEFINED** whether a reset entry to an Exception level will generate an Exception Catch debug event.	0b0

Bits	Name	Description	Reset
[2]	SE2	Controls exception catch on exception entry to Secure EL2. Also controls exception catch on exception return to Secure EL2 in conjunction with EDECCR.SR2. **0b0** If EDECCR.SR2 is 0, then Exception Catch debug events are disabled for Secure EL2. If EDECCR.SR2 is 1, then Exception Catch debug events are enabled for exception returns to Secure EL2. **0b1** If EDECCR.SR2 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL2. If EDECCR.SR2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL2. **Note:** It is **IMPLEMENTATION DEFINED** whether a reset entry to an Exception level will generate an Exception Catch debug event.	0b0
[1]	SE1	Controls exception catch on exception entry to Secure EL1. Also controls exception catch on exception return to Secure EL1 in conjunction with EDECCR.SR1. **0b0** If EDECCR.SR1 is 0, then Exception Catch debug events are disabled for Secure EL1. If EDECCR.SR1 is 1, then Exception Catch debug events are enabled for exception returns to Secure EL1. **0b1** If EDECCR.SR1 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL1. If EDECCR.SR1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL1. **Note:** It is **IMPLEMENTATION DEFINED** whether a reset entry to an Exception level will generate an Exception Catch debug event.	0b0
[0]	SE0	None Access to this field is: **RES0**	x

Accessibility

Component	Offset	Instance	Range
Debug	0x098	EDECCR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.10 OSLAR_EL1, OS Lock Access Register

Used to lock or unlock the OS Lock.

Configurations

If FEAT_Debugv8p2 is not implemented, it is IMPLEMENTATION DEFINED whether external debug accesses to OSLAR_EL1 are ignored and return an error when AllowExternalDebugAccess() returns FALSE for the access.

If FEAT_Debugv8p2 is implemented, external debug accesses to OSLAR_EL1 are ignored and return an error when AllowExternalDebugAccess() returns FALSE for the access.

Attributes

Width

32

Component

Debug

Register offset

0x300

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-117: ext_oslar_el1 bit assignments

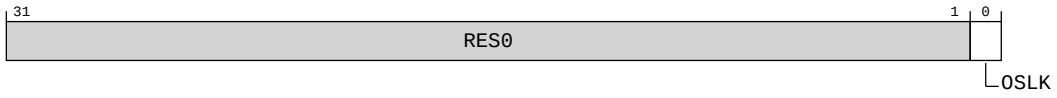


Table B-216: OSLAR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	OSLK	On writes to OSLAR_EL1, bit[0] is copied to the OS Lock. Use ext-EDPRSR.OSLK to check the current status of the lock.	x

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x300	OSLAR_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

WI

When IsCorePowered() && !DoubleLockStatus() && AllowExternalDebugAccess() && ! SoftwareLockStatus()

WO

Otherwise

ERROR

B.5.11 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

Configurations

If FEAT_DoPD is implemented then all fields in this register are in the Core power domain.

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR_EL1.

Attributes

Width

32

Component

Debug

Register offset

0x310

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx0x

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-118: ext_edprcr bit assignments

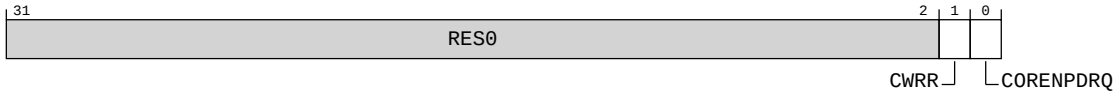


Table B-218: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**
[1]	CWRR	This feature is not supported. Writes to this bit are ignored **0b0** No action. **When OSLockStatus() || SoftwareLockStatus()** Access to this field is: **RAZ/WI** **Otherwise** Access to this field is: WO/**RAZ**	0b0

Bits	Name	Description	Reset
[0]	CORENPDRQ	This field is in the Core power domain, and permitted accesses to this field map to the AArch64-DBGPRCR_EL1.CORENPDRQ field. **0b0** If the system responds to a powerdown request, it powers down Core power domain. **0b1** If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain. **When OSLockStatus()** Access to this field is: **UNKNOWN**/WI **When SoftwareLockStatus()** Access to this field is: RO **Otherwise** Access to this field is: RW	x [5]

Accessibility

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

Component	Offset	Instance	Range
Debug	0x310	EDPRCR	None

This interface is accessible as follows:

When IsCorePowered() && SoftwareLockStatus()
RO

When IsCorePowered() && !SoftwareLockStatus()
RW

Otherwise
ERROR

B.5.12 EDPRSR, External Debug Processor Status Register

Holds information about the reset and powerdown state of the PE.

Configurations

If FEAT_DoPD is implemented then all fields in this register are in the Core power domain.

Attributes

Width
32

[5] On a Cold reset, if the powerup request is implemented and the powerup request has been asserted, this field is an **IMPLEMENTATION DEFINED** choice of 0 or 1. If the powerup request is not asserted, this field is set to 0.

Component

Debug

Register offset

0x314

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xx0x x0x0 xxxx 1x11

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-119: ext_edprsr bit assignments

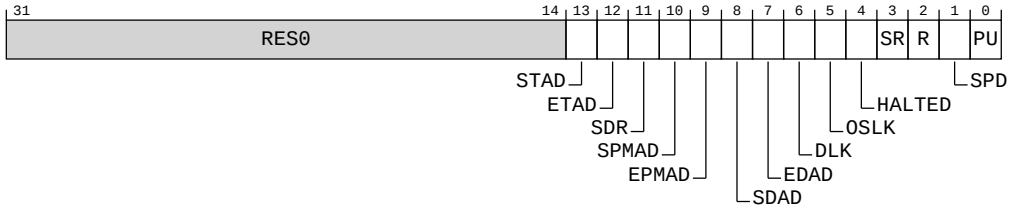


Table B-220: EDPRSR bit descriptions

Bits	Name	Description	Reset
[31:14]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[13]	STAD	Sticky ETAD error. Set to 1 when a Non-secure external debug interface access to an external trace register returns an error because `AllowExternalTraceAccess()` == FALSE for the access. **0b0** Since EDPRSR was last read, no external accesses to the trace unit registers have failed because `AllowExternalTraceAccess()` was FALSE for the access. **0b1** Since EDPRSR was last read, at least one external access to the trace unit registers has failed because `AllowExternalTraceAccess()` was FALSE for the access. If `IsCorePowered()` == TRUE, the Core power domain is powered up, then, following a read of EDPRSR: • If FEAT_DoubleLock is not implemented or DoubleLockStatus() == FALSE then this bit clears to 0. • If FEAT_DoubleLock is implemented and DoubleLockStatus() == TRUE then it is **CONSTRAINED UNPREDICTABLE** whether this bit clears to 0 or is unchanged. This bit is in the Core power domain. **Note:** If FEAT_DoPD is implemented, FEAT_DoubleLock is not implemented. **When DoubleLockStatus() || ext-EDPRSR.R == '1'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RC/**WI**	0b0
[12]	ETAD	External Trace Access Disable status. **0b0** External Non-secure trace unit accesses enabled. `AllowExternalTraceAccess()` == TRUE for a Non-secure access. **0b1** External Non-secure trace unit accesses disabled. `AllowExternalTraceAccess()` == FALSE for a Non-secure access. This bit is in the Core power domain. **Note:** If FEAT_DoPD is implemented, FEAT_DoubleLock is not implemented. **When DoubleLockStatus() || ext-EDPRSR.R == '1'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x

Bits	Name	Description	Reset
[11]	SDR	Sticky Debug Restart. Set to 1 when the PE exits Debug state. Permitted values are: **0b0** The PE has not restarted since EDPRSR was last read. **0b1** The PE has restarted since EDPRSR was last read. **Note:** If a reset occurs when the PE is in Debug state, the PE exits Debug state. SDR is **UNKNOWN** on Warm reset, meaning a debugger must also use the SR bit to determine whether the PE has left Debug state. If the Core power domain is powered up, then following a read of EDPRSR: • This bit clears to 0. This field is in the Core power domain. **When DoubleLockStatus() || ext-EDPRSR.R == '1'** Access to this field is: **UNKNOWN**/WI **When SoftwareLockStatus()** Access to this field is: RO **Otherwise** Access to this field is: RC/**WI**	x
[10]	SPMAD	Sticky EPMAD error. Set to 1 if an external debug interface access to a Performance Monitors register returns an error because `AllowExternalPMUAccess()` == FALSE. Permitted values are: **0b0** No Non-secure external debug interface accesses to the external Performance Monitors registers have failed because `AllowExternalPMUAccess()` == FALSE for the access since EDPRSR was last read. **0b1** At least one Non-secure external debug interface access to the external Performance Monitors register has failed and returned an error because `AllowExternalPMUAccess()` == FALSE for the access since EDPRSR was last read. If the Core power domain is powered up, then following a read of EDPRSR: • This bit clears to 0. This field is in the Core power domain. **When DoubleLockStatus() || ext-EDPRSR.R == '1'** Access to this field is: **UNKNOWN**/WI **When SoftwareLockStatus()** Access to this field is: RO **Otherwise** Access to this field is: RC/**WI**	0b0

Bits	Name	Description	Reset
[9]	EPMAD	External Performance Monitors Non-secure Access Disable status. **0b0** External Non-secure Performance Monitors access enabled. `AllowExternalPMUAccess()` == TRUE for a Non-secure access. **0b1** External Non-secure Performance Monitors access disabled. `AllowExternalPMUAccess()` == FALSE for a Non-secure access. This field is in the Core power domain. **When DoubleLockStatus() || ext-EDPRSR.R == '1'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x
[8]	SDAD	Sticky EDAD error. Set to 1 if an external debug interface access to a debug register returns an error because `AllowExternalDebugAccess()` == FALSE. **0b0** No Non-secure external debug interface accesses to the debug registers have failed because `AllowExternalDebugAccess()` == FALSE for the access since EDPRSR was last read. **0b1** At least one Non-secure external debug interface access to the debug registers has failed and returned an error because `AllowExternalDebugAccess()` == FALSE for the access since EDPRSR was last read. If the Core power domain is powered up, then, following a read of EDPRSR: • This bit clears to 0. This field is in the Core power domain. **When DoubleLockStatus() || ext-EDPRSR.R == '1'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	0b0
[7]	EDAD	External Debug Access Disable status. **0b0** External Non-secure access to breakpoint registers, watchpoint registers, and ext-OSLAR_EL1 enabled. `AllowExternalDebugAccess()` == TRUE for a Non-secure access. **0b1** External Non-secure access to breakpoint registers, watchpoint registers, and ext-OSLAR_EL1 disabled. `AllowExternalDebugAccess()` == FALSE for a Non-secure access. This field is in the Core power domain. **When DoubleLockStatus() || ext-EDPRSR.R == '1'** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x

Bits	Name	Description	Reset
[6]	DLK	This field is **RES0**.	x
[5]	OSLK	OS Lock status bit. A read of this bit returns the value of AArch64-OSLSR_EL1.OSLK. This field is in the Core power domain. Access to this field is: RO	x
[4]	HALTED	Halted status bit. **0b0** PE is in Non-debug state. **0b1** PE is in Debug state. This field is in the Core power domain. Access to this field is: RO	x
[3]	SR	Sticky core Reset status bit. Permitted values are: **0b0** The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read. **0b1** The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read. If EDPRSR.PU reads as 1 and EDPRSR.R reads as 0, which means that the Core power domain is in a powerup state and that the non-debug logic of the PE is not in reset state, then following a read of EDPRSR: • This bit clears to 0. This field is in the Core power domain. **When DoubleLockStatus()** Access to this field is: **UNKNOWN**/WI **When SoftwareLockStatus()** Access to this field is: RO **Otherwise** Access to this field is: RC/**WI**	0b1

Bits	Name	Description	Reset
[2]	R	PE Reset status bit. Permitted values are: **0b0** The non-debug logic of the PE is not in reset state. **0b1** The non-debug logic of the PE is in reset state. This field is in the Core power domain. **When DoubleLockStatus()** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x
[1]	SPD	Sticky core Powerdown status bit. For more information, see *EDPRSR.{DLK, SPD, PU} and the Core power domain* in the *Arm® Architecture Reference Manual for A-profile architecture*. **0b0** If EDPRSR.PU is 0, it is not known whether the state of the debug registers in the Core power domain is lost. If EDPRSR.PU is 1, the state of the debug registers in the Core power domain has not been lost. **0b1** The state of the debug registers in the Core power domain has been lost. If the Core power domain is powered up, then, following a read of EDPRSR: • This bit clears to 0. When *EDPRSR.SPD when the Core domain is in either retention or powerdown state* in the *Arm® Architecture Reference Manual for A-profile architecture*. EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see *EDPRSR.{DLK, SPD, PU} and the Core power domain* in the *Arm® Architecture Reference Manual for A-profile architecture*. This field is in the Core power domain. **When IsCorePowered() && DoubleLockStatus()** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	0b1
[0]	PU	Core powerup status bit. Access to this field is: **RAO/WI**	0b1

Access

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

If the Core power domain is powered up (EDPRSR.PU == 1), then following a read of EDPRSR:

- EDPRSR.{SDR, SPMAD, SDAD, SPD} are cleared to 0.
- EDPRSR.SR is cleared to 0 if the non-debug logic of the PE is not in reset state (EDPRSR.R == 0).

If FEAT_DoPD is not implemented and the Core power domain is powered down (EDPRSR.PU == 0), then:

- EDPRSR.{SDR, SPMAD, SDAD, SR} are all **UNKNOWN**, and are either reset or restored on being powered up.
- EDPRSR.SPD is not cleared following a read of EDPRSR. See the SPD bit description for more information.

The clearing of bits is an indirect write to EDPRSR.

Accessibility

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

If the Core power domain is powered up (EDPRSR.PU == 1), then following a read of EDPRSR:

- EDPRSR.{SDR, SPMAD, SDAD, SPD} are cleared to 0.
- EDPRSR.SR is cleared to 0 if the non-debug logic of the PE is not in reset state (EDPRSR.R == 0).

If FEAT_DoPD is not implemented and the Core power domain is powered down (EDPRSR.PU == 0), then:

- EDPRSR.{SDR, SPMAD, SDAD, SR} are all UNKNOWN, and are either reset or restored on being powered up.
- EDPRSR.SPD is not cleared following a read of EDPRSR. See the SPD bit description for more information.

The clearing of bits is an indirect write to EDPRSR.

Component	Offset	Instance	Range
Debug	0x314	EDPRSR	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.13 DBGBVR0_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register ext-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of ext-DBGBCR<n>_EL1.BT.

- When ext-DBGBCR<n>_EL1.BT is 0b0x0x, this register holds a virtual address.
- When ext-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When ext-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of ext-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

64

Component

Debug

Register offset

0x400

Access type

See bit descriptions

Reset value

When ext-DBGBCR0_EL1.BT == '0x0x'

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

When ext-DBGBCR0_EL1.BT == '001x'

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

When ext-DBGBCR0_EL1.BT == '011x'

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

When ext-DBGBCR0_EL1.BT == '100x'

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

When ext-DBGBCR0_EL1.BT == '101x'

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

When ext-DBGBCR0_EL1.BT == '110x'

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

When ext-DBGBCR0_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When ext-DBGBCR0_EL1.BT == '0x0'

Figure B-120: ext_dbgbvr0_el1 bit assignments

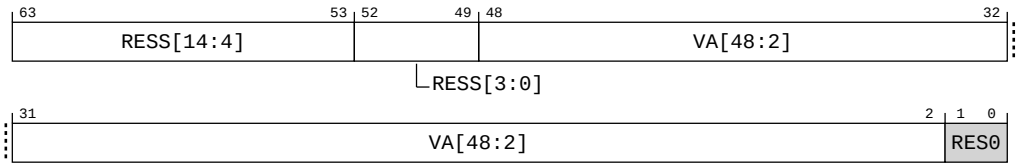


Table B-222: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	If the address is being matched in an AArch64 stage 1 translation regime: • This field contains bits[48:2] of the address for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR0_EL1.BT == '001'

Figure B-121: ext_dbgbvr0_el1 bit assignments

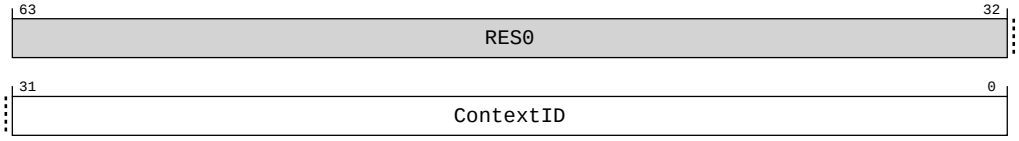


Table B-223: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), EL2 is using AArch64, AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against the following: • AArch64-CONTEXTIDR_EL1 when the PE is executing at AArch64.	32{x}

When ext-DBGBCR0_EL1.BT == '011'

Figure B-122: ext_dbgbvr0_el1 bit assignments

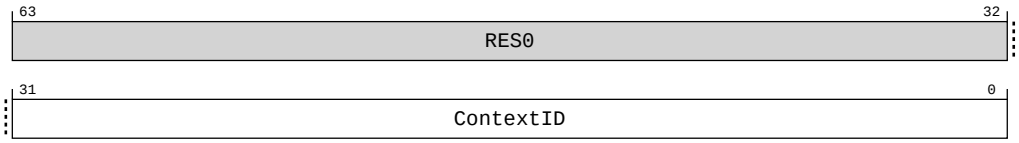


Table B-224: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR0_EL1.BT == '100'

Figure B-123: ext_dbgbvr0_el1 bit assignments

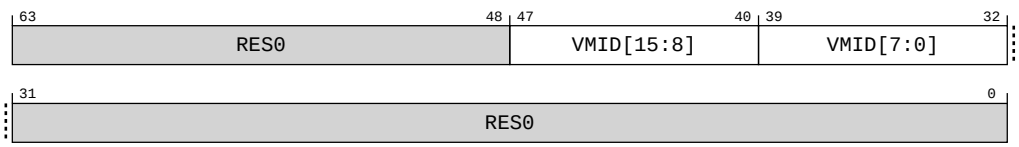


Table B-225: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR0_EL1.BT == '101'

Figure B-124: ext_dbgbvr0_el1 bit assignments

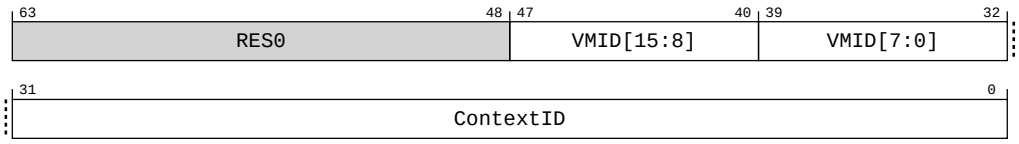


Table B-226: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR0_EL1.BT == '110'

Figure B-125: ext_dbgbvr0_el1 bit assignments

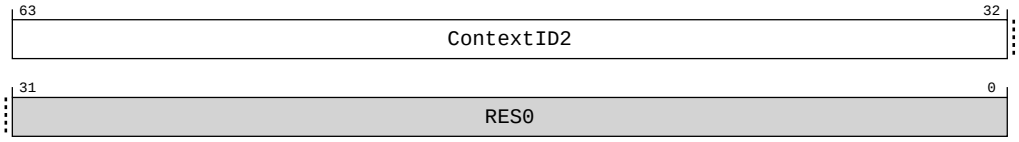


Table B-227: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR0_EL1.BT == '111'

Figure B-126: ext_dbgbvr0_el1 bit assignments

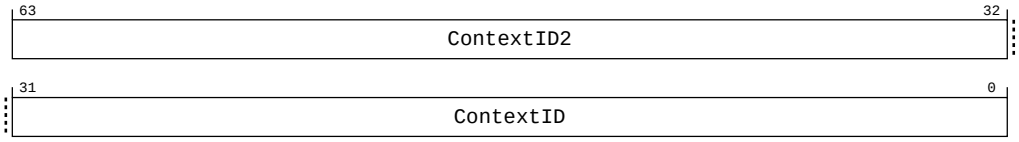


Table B-228: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x400	DBGBVR0_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.14 DBGBCR0_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register ext-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

32

Component

Debug

Register offset

0x408

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-127: ext_dbgbcr0_el1 bit assignments

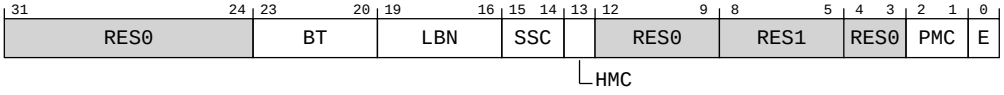


Table B-230: DBGBCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000 but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value. **0b0011** As 0b0010, with linking enabled. **0b0100** Unlinked instruction address mismatch. ext-DBGBVR<n>_EL1 is the address of an instruction to be stepped. **0b0101** As 0b0100, but linked to a Context matching breakpoint. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. ext-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. ext-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. Constraints on breakpoint programming mean some values are reserved under certain conditions. For more information on the operation of the SSC, HMC, and PMC fields, and on the effect of programming this field to a reserved value, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture* and *Reserved DBGBCR0_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable breakpoint ext-DBGBVR<n>_EL1. Possible values are: **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x408	DBGBCR0_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.15 DBGBVR1_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register ext-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of ext-DBGBCR<n>_EL1.BT.

- When ext-DBGBCR<n>_EL1.BT is 0b0x0x, this register holds a virtual address.
- When ext-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When ext-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of ext-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

64

Component

Debug

Register offset

0x410

Access type

See bit descriptions

Reset value

When ext-DBGBCR1_EL1.BT == '0x0x'

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

When ext-DBGBCR1_EL1.BT == '001x'

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

When ext-DBGBCR1_EL1.BT == '011x'

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

When ext-DBGBCR1_EL1.BT == '100x'

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

When ext-DBGBCR1_EL1.BT == '101x'

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

When ext-DBGBCR1_EL1.BT == '110x'

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

When ext-DBGBCR1_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When ext-DBGBCR1_EL1.BT == '0x0'

Figure B-128: ext_dbgbvr1_el1 bit assignments

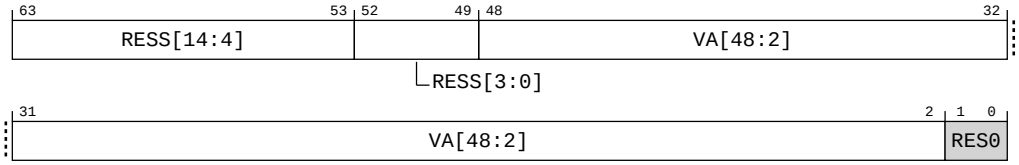


Table B-232: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	`11{x}`

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	If the address is being matched in an AArch64 stage 1 translation regime: • This field contains bits[48:2] of the address for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR1_EL1.BT == '001'

Figure B-129: ext_dbgbvr1_el1 bit assignments

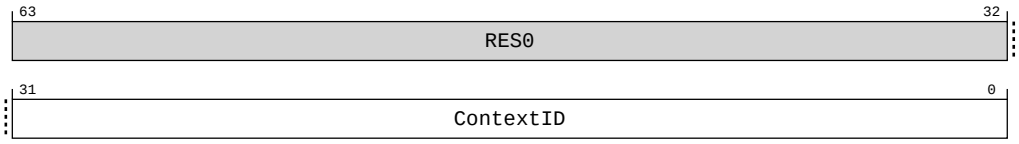


Table B-233: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), EL2 is using AArch64, AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against the following: • AArch64-CONTEXTIDR_EL1 when the PE is executing at AArch64.	32{x}

When ext-DBGBCR1_EL1.BT == '011'

Figure B-130: ext_dbgbvr1_el1 bit assignments

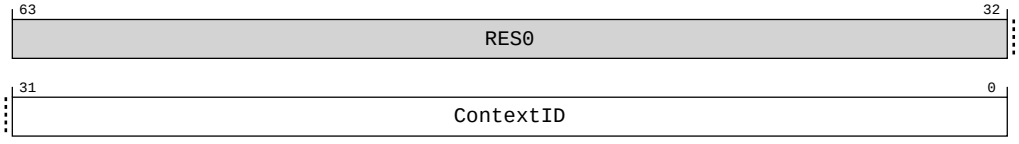


Table B-234: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR1_EL1.BT == '100'

Figure B-131: ext_dbgbvr1_el1 bit assignments

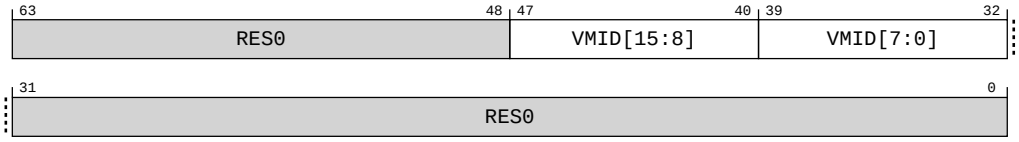


Table B-235: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR1_EL1.BT == '101'

Figure B-132: ext_dbgbvr1_el1 bit assignments

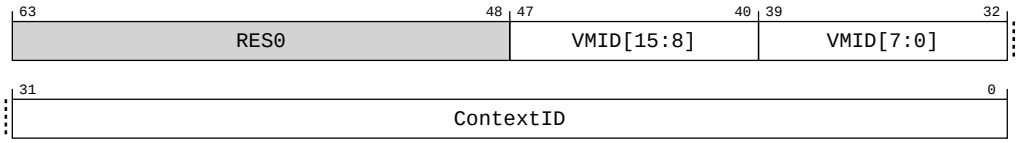


Table B-236: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR1_EL1.BT == '110'

Figure B-133: ext_dbgbvr1_el1 bit assignments

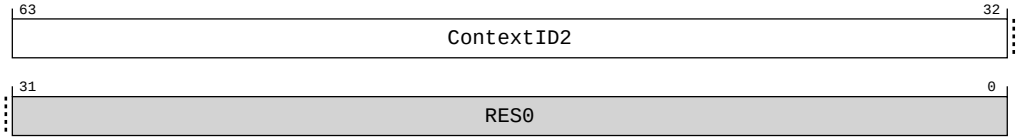


Table B-237: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR1_EL1.BT == '111'

Figure B-134: ext_dbgbvr1_el1 bit assignments

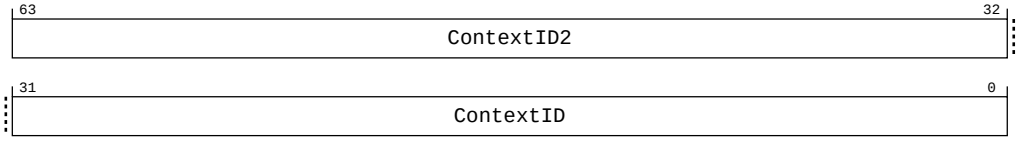


Table B-238: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x410	DBGBVR1_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.16 DBGBCR1_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register ext-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

32

Component

Debug

Register offset

0x418

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-135: ext_dbgbcr1_el1 bit assignments

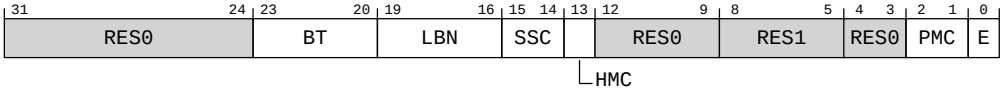


Table B-240: DBGBCR1_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000 but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value. **0b0011** As 0b0010, with linking enabled. **0b0100** Unlinked instruction address mismatch. ext-DBGBVR<n>_EL1 is the address of an instruction to be stepped. **0b0101** As 0b0100, but linked to a Context matching breakpoint. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. ext-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. ext-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. Constraints on breakpoint programming mean some values are reserved under certain conditions. For more information on the operation of the SSC, HMC, and PMC fields, and on the effect of programming this field to a reserved value, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture* and *Reserved DBGBCR1_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable breakpoint ext-DBGBVR<n>_EL1. Possible values are: **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x418	DBGBCR1_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.17 DBGBVR2_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register ext-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of ext-DBGBCR<n>_EL1.BT.

- When ext-DBGBCR<n>_EL1.BT is 0b0x0x, this register holds a virtual address.
- When ext-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When ext-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of ext-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

64

Component

Debug

Register offset

0x420

Access type

See bit descriptions

Reset value

When ext-DBGBCR2_EL1.BT == '0x0x'

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

When ext-DBGBCR2_EL1.BT == '001x'

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

When ext-DBGBCR2_EL1.BT == '011x'

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

When ext-DBGBCR2_EL1.BT == '100x'

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

When ext-DBGBCR2_EL1.BT == '101x'

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

When ext-DBGBCR2_EL1.BT == '110x'

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

When ext-DBGBCR2_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When ext-DBGBCR2_EL1.BT == '0x0'

Figure B-136: ext_dbgbvr2_el1 bit assignments

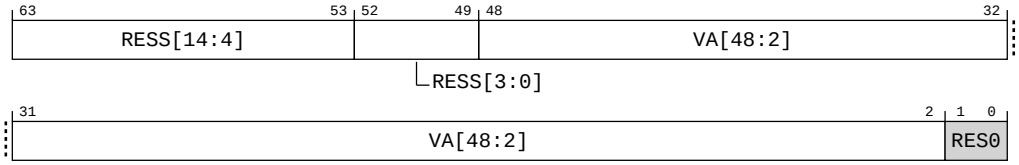


Table B-242: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	`11{x}`

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	If the address is being matched in an AArch64 stage 1 translation regime: • This field contains bits[48:2] of the address for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR2_EL1.BT == '001'

Figure B-137: ext_dbgbvr2_el1 bit assignments

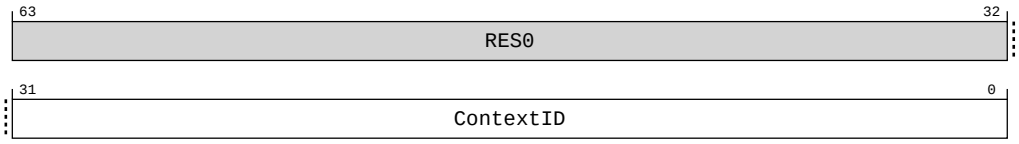


Table B-243: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), EL2 is using AArch64, AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against the following: • AArch64-CONTEXTIDR_EL1 when the PE is executing at AArch64.	32{x}

When ext-DBGBCR2_EL1.BT == '011'

Figure B-138: ext_dbgbvr2_el1 bit assignments

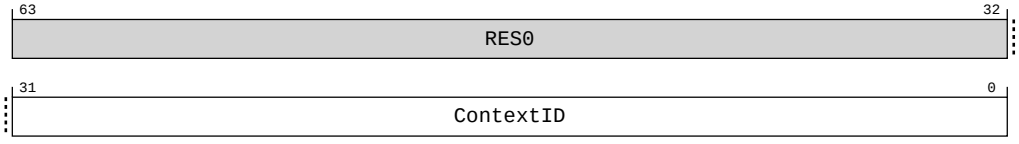


Table B-244: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR2_EL1.BT == '100'

Figure B-139: ext_dbgbvr2_el1 bit assignments

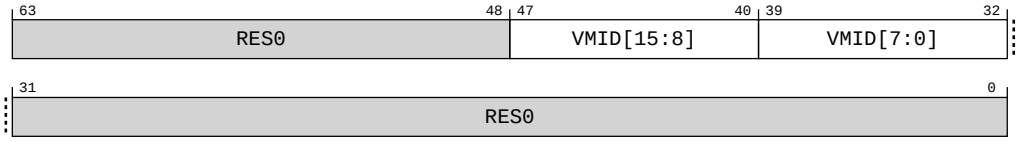


Table B-245: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR2_EL1.BT == '101'

Figure B-140: ext_dbgbvr2_el1 bit assignments

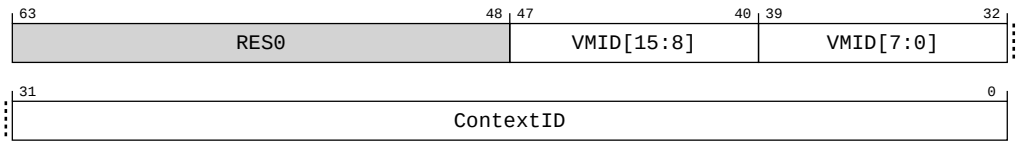


Table B-246: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR2_EL1.BT == '110'

Figure B-141: ext_dbgbvr2_el1 bit assignments

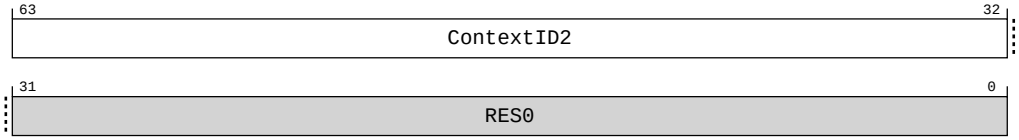


Table B-247: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR2_EL1.BT == '111'

Figure B-142: ext_dbgbvr2_el1 bit assignments

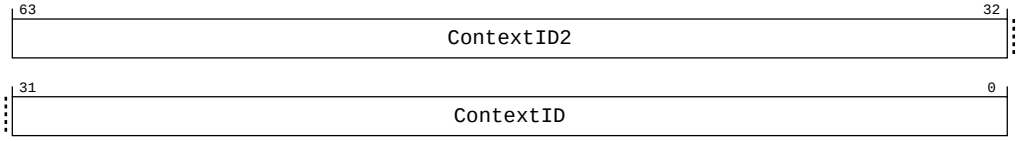


Table B-248: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x420	DBGBVR2_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.18 DBGBCR2_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register ext-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

32

Component

Debug

Register offset

0x428

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-143: ext_dbgbcr2_el1 bit assignments

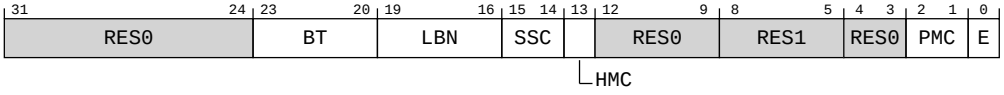


Table B-250: DBGBCR2_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000 but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value. **0b0011** As 0b0010, with linking enabled. **0b0100** Unlinked instruction address mismatch. ext-DBGBVR<n>_EL1 is the address of an instruction to be stepped. **0b0101** As 0b0100, but linked to a Context matching breakpoint. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. ext-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. ext-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. Constraints on breakpoint programming mean some values are reserved under certain conditions. For more information on the operation of the SSC, HMC, and PMC fields, and on the effect of programming this field to a reserved value, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture* and *Reserved DBGBCR2_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable breakpoint ext-DBGBVR<n>_EL1. Possible values are: **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x428	DBGBCR2_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.19 DBGBVR3_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register ext-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of ext-DBGBCR<n>_EL1.BT.

- When ext-DBGBCR<n>_EL1.BT is 0b0x0x, this register holds a virtual address.
- When ext-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When ext-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of ext-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

64

Component

Debug

Register offset

0x430

Access type

See bit descriptions

Reset value

When ext-DBGBCR3_EL1.BT == '0x0x'

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

When ext-DBGBCR3_EL1.BT == '001x'

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

When ext-DBGBCR3_EL1.BT == '011x'

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

When ext-DBGBCR3_EL1.BT == '100x'

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

When ext-DBGBCR3_EL1.BT == '101x'

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

When ext-DBGBCR3_EL1.BT == '110x'

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

When ext-DBGBCR3_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When ext-DBGBCR3_EL1.BT == '0x0'

Figure B-144: ext_dbgbvr3_el1 bit assignments

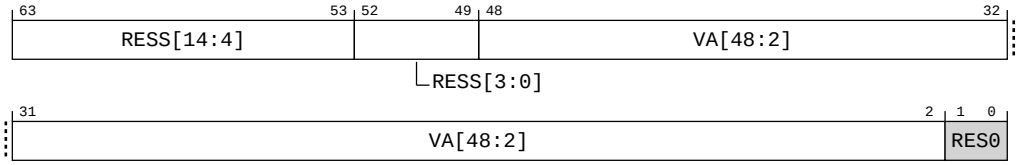


Table B-252: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	`11{x}`

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	If the address is being matched in an AArch64 stage 1 translation regime: • This field contains bits[48:2] of the address for comparison.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR3_EL1.BT == '001'

Figure B-145: ext_dbgbvr3_el1 bit assignments

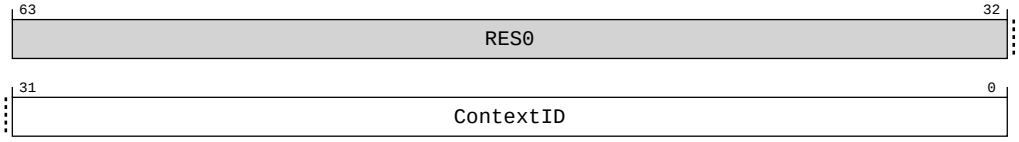


Table B-253: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), EL2 is using AArch64, AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against the following: • AArch64-CONTEXTIDR_EL1 when the PE is executing at AArch64.	32{x}

When ext-DBGBCR3_EL1.BT == '011'

Figure B-146: ext_dbgbvr3_el1 bit assignments

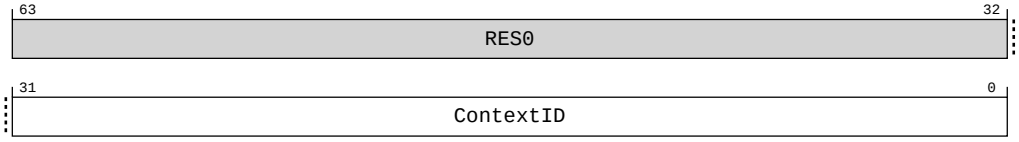


Table B-254: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR3_EL1.BT == '100'

Figure B-147: ext_dbgbvr3_el1 bit assignments

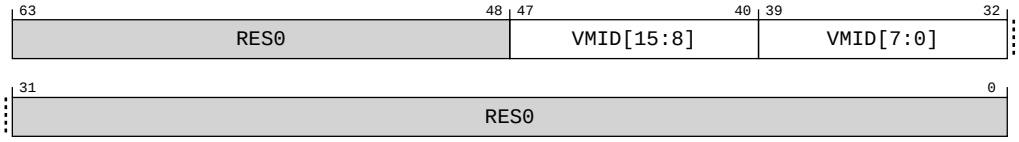


Table B-255: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR3_EL1.BT == '101'

Figure B-148: ext_dbgbvr3_el1 bit assignments

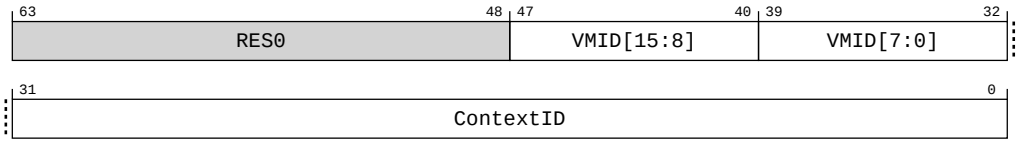


Table B-256: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR3_EL1.BT == '110'

Figure B-149: ext_dbgbvr3_el1 bit assignments

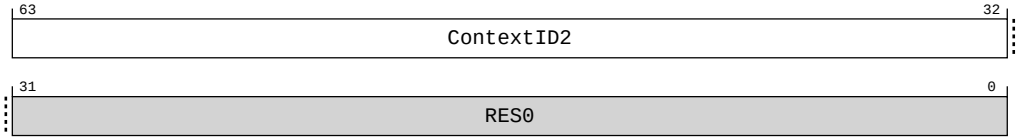


Table B-257: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR3_EL1.BT == '111'

Figure B-150: ext_dbgbvr3_el1 bit assignments

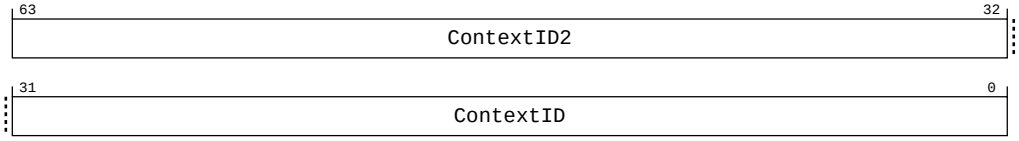


Table B-258: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x430	DBGBVR3_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.20 DBGBCR3_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register ext-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

32

Component

Debug

Register offset

0x438

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-151: ext_dbgbcr3_el1 bit assignments

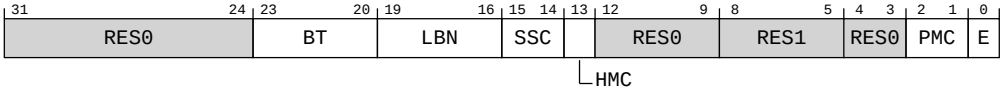


Table B-260: DBGBCR3_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000 but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value. **0b0011** As 0b0010, with linking enabled. **0b0100** Unlinked instruction address mismatch. ext-DBGBVR<n>_EL1 is the address of an instruction to be stepped. **0b0101** As 0b0100, but linked to a Context matching breakpoint. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. ext-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. ext-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. Constraints on breakpoint programming mean some values are reserved under certain conditions. For more information on the operation of the SSC, HMC, and PMC fields, and on the effect of programming this field to a reserved value, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture* and *Reserved DBGBCR3_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable breakpoint ext-DBGBVR<n>_EL1. Possible values are: **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x438	DBGBCR3_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.21 DBGBVR4_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register ext-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of ext-DBGBCR<n>_EL1.BT.

- When ext-DBGBCR<n>_EL1.BT is 0b0x0x, this register holds a virtual address.
- When ext-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When ext-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of ext-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

64

Component

Debug

Register offset

0x440

Access type

See bit descriptions

Reset value

When ext-DBGBCR4_EL1.BT == '0x0x'

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

When ext-DBGBCR4_EL1.BT == '001x'

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

When ext-DBGBCR4_EL1.BT == '011x'

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

When ext-DBGBCR4_EL1.BT == '100x'

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

When ext-DBGBCR4_EL1.BT == '101x'

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

When ext-DBGBCR4_EL1.BT == '110x'

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

When ext-DBGBCR4_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When ext-DBGBCR4_EL1.BT == '0x0'

Figure B-152: ext_dbgbvr4_el1 bit assignments

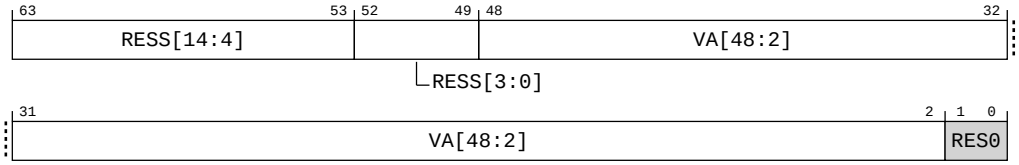


Table B-262: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	`11{x}`

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	If the address is being matched in an AArch64 stage 1 translation regime: • This field contains bits[48:2] of the address for comparison.	47{x}
[1:0]	RES0	Reserved	RES0

When ext-DBGBCR4_EL1.BT == '001'

Figure B-153: ext_dbgbvr4_el1 bit assignments

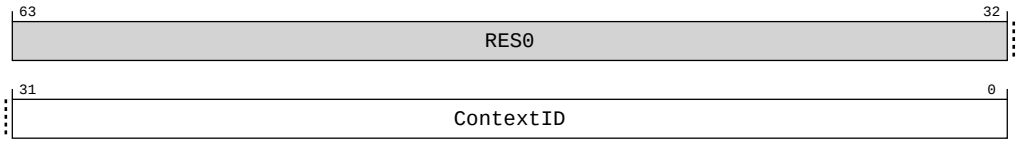


Table B-263: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), EL2 is using AArch64, AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against the following: • AArch64-CONTEXTIDR_EL1 when the PE is executing at AArch64.	32{x}

When ext-DBGBCR4_EL1.BT == '011'

Figure B-154: ext_dbgbvr4_el1 bit assignments

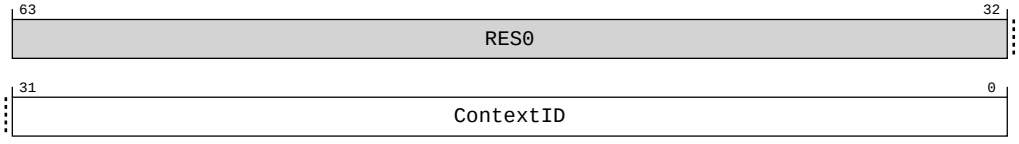


Table B-264: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR4_EL1.BT == '100'

Figure B-155: ext_dbgbvr4_el1 bit assignments

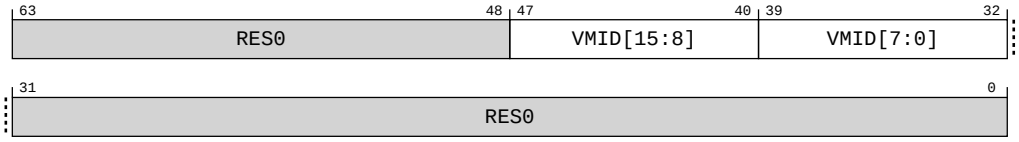


Table B-265: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR4_EL1.BT == '101'

Figure B-156: ext_dbgbvr4_el1 bit assignments

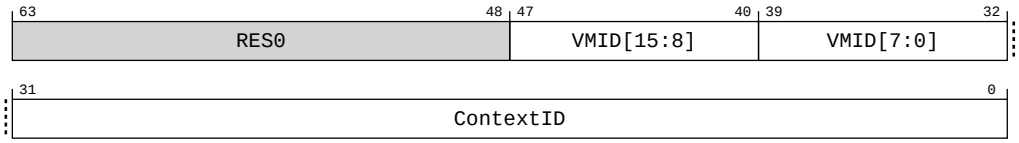


Table B-266: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR4_EL1.BT == '110'

Figure B-157: ext_dbgbvr4_el1 bit assignments

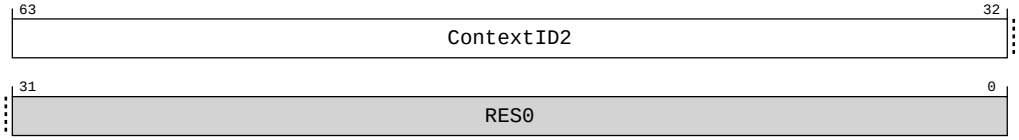


Table B-267: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR4_EL1.BT == '111'

Figure B-158: ext_dbgbvr4_el1 bit assignments

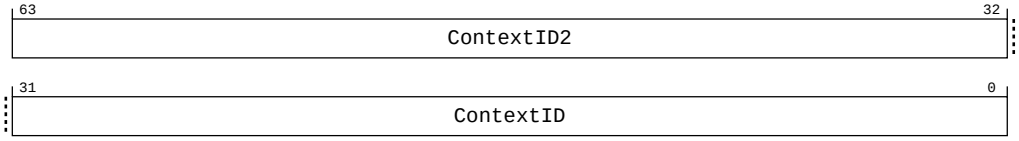


Table B-268: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x440	DBGBVR4_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.22 DBGBCR4_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register ext-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

32

Component

Debug

Register offset

0x448

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-159: ext_dbgbcr4_el1 bit assignments

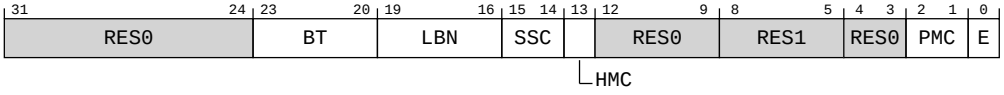


Table B-270: DBGBCR4_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000 but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value. **0b0011** As 0b0010, with linking enabled. **0b0100** Unlinked instruction address mismatch. ext-DBGBVR<n>_EL1 is the address of an instruction to be stepped. **0b0101** As 0b0100, but linked to a Context matching breakpoint. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. ext-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. ext-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. Constraints on breakpoint programming mean some values are reserved under certain conditions. For more information on the operation of the SSC, HMC, and PMC fields, and on the effect of programming this field to a reserved value, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture* and *Reserved DBGBCR4_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable breakpoint ext-DBGBVR<n>_EL1. Possible values are: **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x448	DBGBCR4_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.23 DBGBVR5_EL1, Debug Breakpoint Value Registers

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register ext-DBGBCR<n>_EL1.

Configurations

How this register is interpreted depends on the value of ext-DBGBCR<n>_EL1.BT.

- When ext-DBGBCR<n>_EL1.BT is 0b0x0x, this register holds a virtual address.
- When ext-DBGBCR<n>_EL1.BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b100x, this register holds a VMID.
- When ext-DBGBCR<n>_EL1.BT is 0b101x, this register holds a VMID and a Context ID.
- When ext-DBGBCR<n>_EL1.BT is 0b111x, this register holds two Context ID values.

For other values of ext-DBGBCR<n>_EL1.BT, this register is RES0.

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

64

Component

Debug

Register offset

0x450

Access type

See bit descriptions

Reset value

When ext-DBGBCR5_EL1.BT == '0x0x'

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

When ext-DBGBCR5_EL1.BT == '001x'

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

When ext-DBGBCR5_EL1.BT == '011x'

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

When ext-DBGBCR5_EL1.BT == '100x'

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

When ext-DBGBCR5_EL1.BT == '101x'

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

When ext-DBGBCR5_EL1.BT == '110x'

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

When ext-DBGBCR5_EL1.BT == '111x'

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When ext-DBGBCR5_EL1.BT == '0x0'

Figure B-160: ext_dbgbvr5_el1 bit assignments

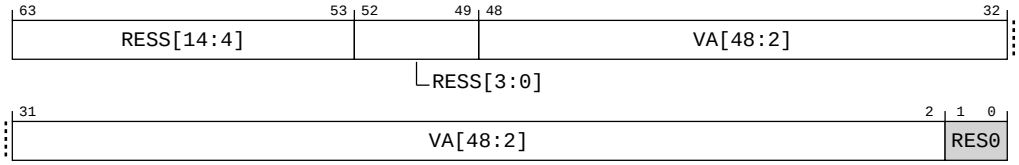


Table B-272: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	`11{x}`

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	If the address is being matched in an AArch64 stage 1 translation regime: • This field contains bits[48:2] of the address for comparison.	47{x}
[1:0]	RES0	Reserved	RES0

When ext-DBGBCR5_EL1.BT == '001'

Figure B-161: ext_dbgbvr5_el1 bit assignments

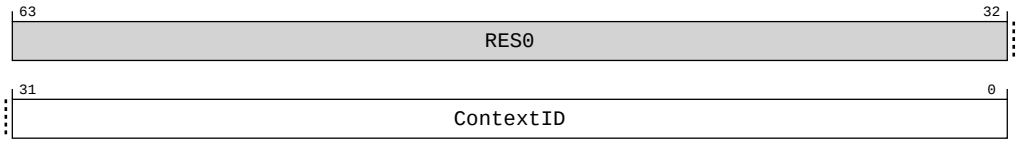


Table B-273: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), EL2 is using AArch64, AArch64-HCR_EL2.E2H is 1, and either: • The PE is executing at EL2. • AArch64-HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state. Otherwise, the value is compared against the following: • AArch64-CONTEXTIDR_EL1 when the PE is executing at AArch64.	32{x}

When ext-DBGBCR5_EL1.BT == '011'

Figure B-162: ext_dbgbvr5_el1 bit assignments

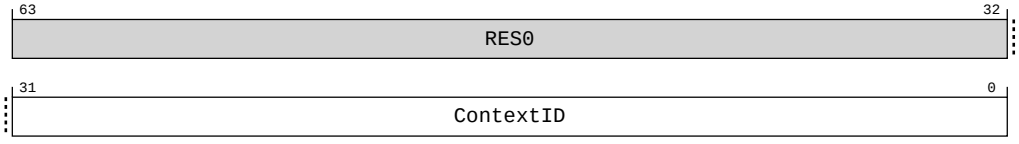


Table B-274: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR5_EL1.BT == '100'

Figure B-163: ext_dbgbvr5_el1 bit assignments

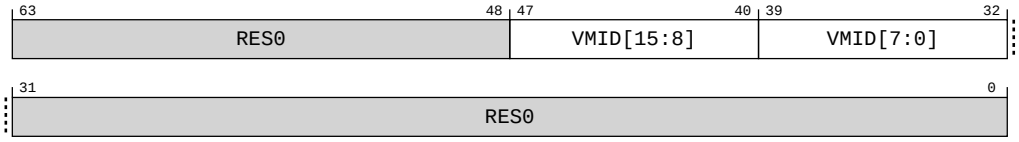


Table B-275: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR5_EL1.BT == '101'

Figure B-164: ext_dbgbvr5_el1 bit assignments

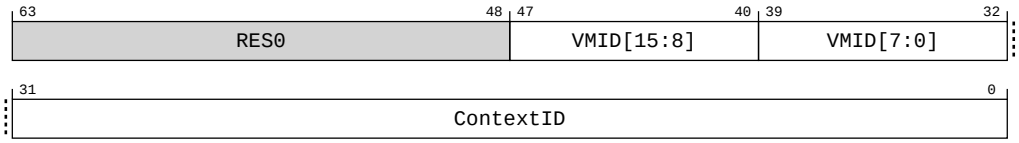


Table B-276: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:40]	VMID[15:8]	**When AArch64-VTCR_EL2.VS == '1'** Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0]. **Otherwise** RES0	8{x}
[39:32]	VMID[7:0]	VMID value for comparison. The VMID is 8 bits when any of the following are true: • AArch64-VTCR_EL2.VS is 0. • FEAT_VMID16 is not implemented.	8{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

When ext-DBGBCR5_EL1.BT == '110'

Figure B-165: ext_dbgbvr5_el1 bit assignments

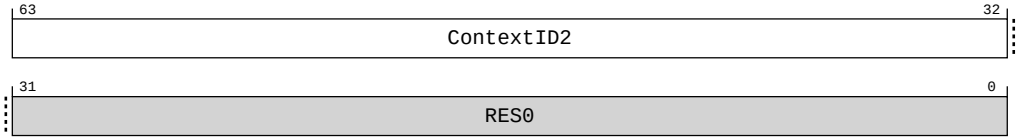


Table B-277: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	**RES0**	Reserved	**RES0**

When ext-DBGBCR5_EL1.BT == '111'

Figure B-166: ext_dbgbvr5_el1 bit assignments

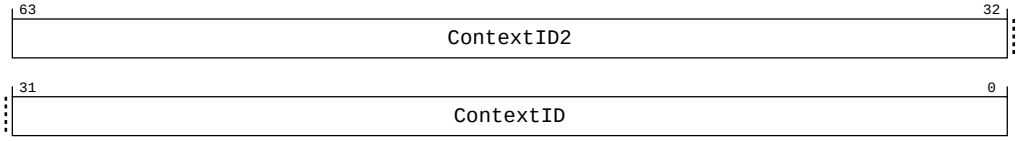


Table B-278: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32{x}

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x450	DBGBVR5_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.24 DBGBCR5_EL1, Debug Breakpoint Control Registers

Holds control information for a breakpoint. Forms breakpoint n together with value register ext-DBGBVR<n>_EL1.

Configurations

If breakpoint n is not implemented then accesses to this register are:

- RES0 when IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess().
- A CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR otherwise.

Attributes

Width

32

Component

Debug

Register offset

0x458

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-167: ext_dbgbcr5_el1 bit assignments

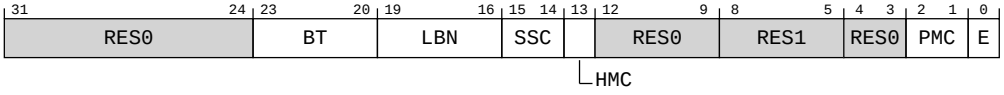


Table B-280: DBGBCR5_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. Possible values are: **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** As 0b0000 but linked to a Context matching breakpoint. **0b0010** Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, and the Effective value of AArch64-HCR_EL2.E2H is 1, if either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value. **0b0011** As 0b0010, with linking enabled. **0b0100** Unlinked instruction address mismatch. ext-DBGBVR<n>_EL1 is the address of an instruction to be stepped. **0b0101** As 0b0100, but linked to a Context matching breakpoint. **0b0110** Unlinked AArch64-CONTEXTIDR_EL1 match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1. **0b0111** As 0b0110, with linking enabled. **0b1000** Unlinked VMID match. ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1001** As 0b1000, with linking enabled. **0b1010** Unlinked VMID and Context ID match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID. **0b1011** As 0b1010, with linking enabled. **0b1100** Unlinked AArch64-CONTEXTIDR_EL2 match. ext-DBGBVR<n>_EL1.ContextID2 is a Context ID compared against AArch64-CONTEXTIDR_EL2. **0b1101** As 0b1100, with linking enabled. **0b1110** Unlinked Full Context ID match. ext-DBGBVR<n>_EL1.ContextID is compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.ContextID2 is compared against AArch64-CONTEXTIDR_EL2. **0b1111** As 0b1110, with linking enabled. Constraints on breakpoint programming mean some values are reserved under certain conditions. For more information on the operation of the SSC, HMC, and PMC fields, and on the effect of programming this field to a reserved value, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture* and *Reserved DBGBCR5_EL1.BT values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to. For all other breakpoint types this field is ignored and reads of the register return an **UNKNOWN** value. This field is ignored when the value of DBGBCR<n>_EL1.E is 0.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see *Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values* in the *Arm® Architecture Reference Manual for A-profile architecture*. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x
[12:9]	**RES0**	Reserved	**RES0**
[8:5]	**RES1**	Reserved	**RES1**
[4:3]	**RES0**	Reserved	**RES0**
[2:1]	PMC	Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the ext-DBGBCR<n>_EL1.SSC description. For more information on the operation of the SSC, HMC, and PMC fields, see *Execution conditions for which a breakpoint generates Breakpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable breakpoint ext-DBGBVR<n>_EL1. Possible values are: **0b0** Breakpoint disabled. **0b1** Breakpoint enabled.	x

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x458	DBGBCR5_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.25 DBGWVR0_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register ext-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

64

Component

Debug

Register offset

0x800

Access type

See bit descriptions

Reset value

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


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-168: ext_dbgwvr0_el1 bit assignments

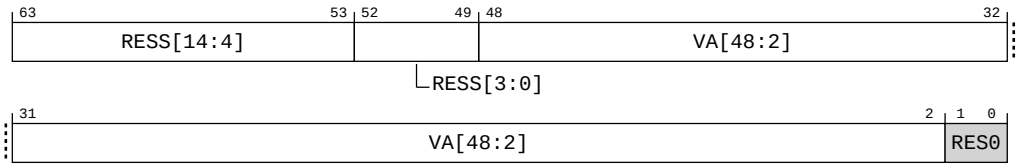


Table B-282: DBGWVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Hardware and software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	`11{x}`
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	`xxxx`
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1.	`47{x}`
[1:0]	**RES0**	Reserved	**RES0**

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x800	DBGWVR0_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.26 DBGWCR0_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register ext-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

32

Component

Debug

Register offset

0x808

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-169: ext_dbgwcr0_el1 bit assignments

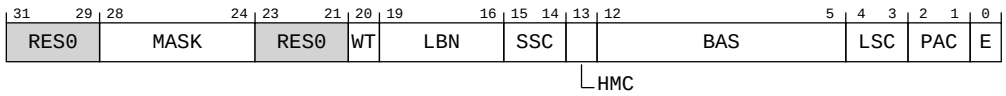


Table B-284: DBGWCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[31:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	5{x}
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	x
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	xxxx
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	x

Bits	Name	Description	Reset
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by ext-DBGWVR<n>_EL1 is being watched. Table B-285: BAS description table 1 on page 1250 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-286: BAS description table 2 on page 1250 If ext-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table B-285: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table B-286: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x808	DBGWCR0_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.27 DBGWVR1_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register ext-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

64

Component

Debug

Register offset

0x810

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-170: ext_dbgwvr1_el1 bit assignments

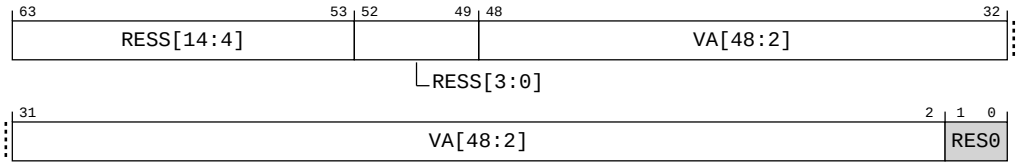


Table B-288: DBGWVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Hardware and software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x810	DBGWVR1_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.28 DBGWCR1_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register ext-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

32

Component

Debug

Register offset

0x818

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-171: ext_dbgwcr1_el1 bit assignments

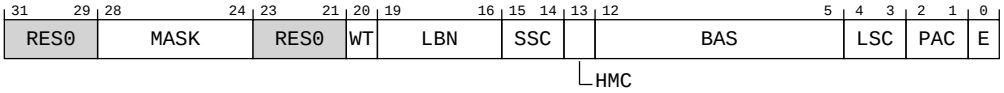


Table B-290: DBGWCR1_EL1 bit descriptions

Bits	Name	Description	Reset
[31:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	`5{x}`
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	`x`
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	`xxxx`
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	`xx`
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	`x`

Bits	Name	Description	Reset
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by ext-DBGWVR<n>_EL1 is being watched. Table B-291: BAS description table 1 on page 1255 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-292: BAS description table 2 on page 1255 If ext-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table B-291: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table B-292: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x818	DBGWCR1_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.29 DBGWVR2_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register ext-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

64

Component

Debug

Register offset

0x820

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-172: ext_dbgwvr2_el1 bit assignments

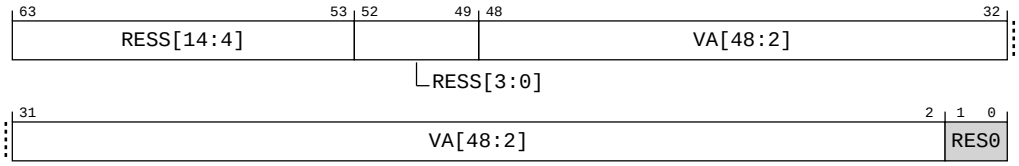


Table B-294: DBGWVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Hardware and software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x820	DBGWVR2_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.30 DBGWCR2_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register ext-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

32

Component

Debug

Register offset

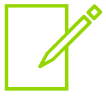
0x828

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-173: ext_dbgwcr2_el1 bit assignments

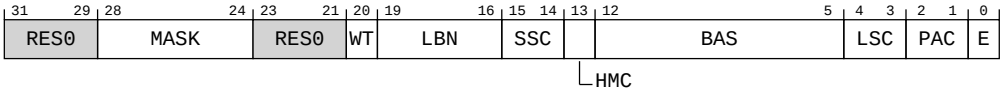


Table B-296: DBGWCR2_EL1 bit descriptions

Bits	Name	Description	Reset
[31:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	`5{x}`
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	`x`
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	`xxxx`
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	`xx`
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	`x`

Bits	Name	Description	Reset
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by ext-DBGWVR<n>_EL1 is being watched. Table B-297: BAS description table 1 on page 1260 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-298: BAS description table 2 on page 1260 If ext-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table B-297: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table B-298: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x828	DBGWCR2_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.31 DBGWVR3_EL1, Debug Watchpoint Value Registers

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register ext-DBGWCR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

64

Component

Debug

Register offset

0x830

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-174: ext_dbgwvr3_el1 bit assignments

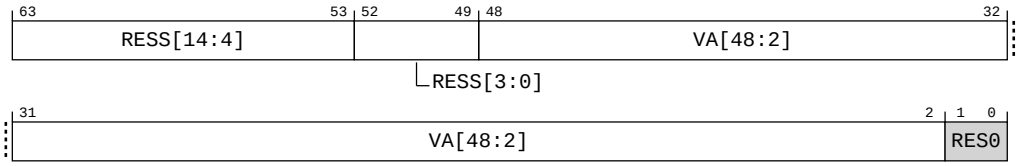


Table B-300: DBGWVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RESS[14:4]	Reserved, Sign extended. Hardware and software must treat this field as **RES0** if the most significant bit of VA is 0 or **RES0**, and as RES1 if the most significant bit of VA is 1. Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether: • The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value. • The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.	11{x}
[52:49]	RESS[3:0]	Extension to RESS[14:4]. For more information, see RESS[14:4].	xxxx
[48:2]	VA[48:2]	Bits[48:2] of the address value for comparison. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1.	47{x}
[1:0]	**RES0**	Reserved	**RES0**

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x830	DBGWVR3_EL1	63:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.32 DBGWCR3_EL1, Debug Watchpoint Control Registers

Holds control information for a watchpoint. Forms watchpoint n together with value register ext-DBGWVR<n>_EL1.

Configurations

If watchpoint n is not implemented then accesses to this register are:

- When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess(), RES0.
- Otherwise, a CONSTRAINED UNPREDICTABLE choice of RES0 or ERROR.

Attributes

Width

32

Component

Debug

Register offset

0x838

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-175: ext_dbgwcr3_el1 bit assignments

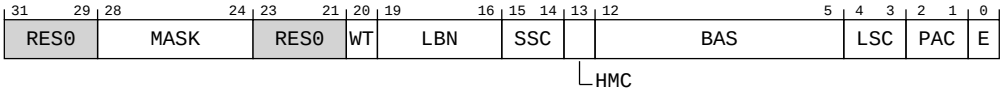


Table B-302: DBGWCR3_EL1 bit descriptions

Bits	Name	Description	Reset
[31:29]	**RES0**	Reserved	**RES0**
[28:24]	MASK	Address mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask. **0b00001** Reserved. **0b00010** Reserved. If programmed with a reserved value, a watchpoint must behave as if either: • MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1. • The watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture. Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).	`5{x}`
[23:21]	**RES0**	Reserved	**RES0**
[20]	WT	Watchpoint type. Possible values are: **0b0** Unlinked data address match. **0b1** Linked data address match.	`x`
[19:16]	LBN	Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.	`xxxx`
[15:14]	SSC	Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	`xx`
[13]	HMC	Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	`x`

Bits	Name	Description	Reset
[12:5]	BAS	Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by ext-DBGWVR<n>_EL1 is being watched. Table B-303: BAS description table 1 on page 1265 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-304: BAS description table 2 on page 1265 If ext-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1. The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See *Reserved DBGWCR<n>.BAS values* in the *Arm® Architecture Reference Manual for A-profile architecture*.	8{x}
[4:3]	LSC	Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are: **0b01** Match instructions that load from a watchpointed address. **0b10** Match instructions that store to a watchpointed address. **0b11** Match instructions that load from or store to a watchpointed address. All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.	xx
[2:1]	PAC	Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields. For more information on the operation of the SSC, HMC, and PAC fields, see *Execution conditions for which a watchpoint generates Watchpoint exceptions* in the *Arm® Architecture Reference Manual for A-profile architecture*.	xx
[0]	E	Enable watchpoint n. Possible values are: **0b0** Watchpoint disabled. **0b1** Watchpoint enabled.	x

Table B-303: BAS description table 1

BAS	Description
xxxxxxx1	Match byte at AArch64-DBGWVR<n>_EL1
xxxxxx1x	Match byte at AArch64-DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at AArch64-DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table B-304: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Access

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Accessibility

SoftwareLockStatus() depends on the type of access attempted and AllowExternalDebugAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

Component	Offset	Instance	Range
Debug	0x838	DBGWCR3_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && AllowExternalDebugAccess() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.33 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

Access type

See bit descriptions

Reset value

0100 0001 0001 1111 1101 0100 1110 0010

Bit descriptions

Figure B-176: ext_midr_el1 bit assignments

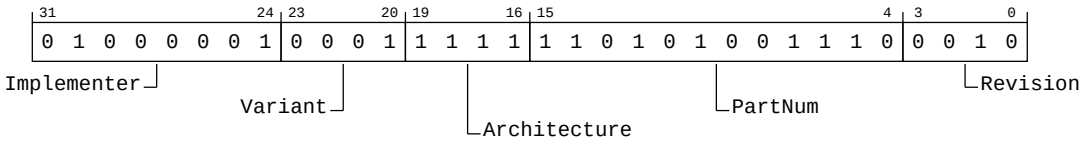


Table B-306: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	Implementer	Indicates the implementer code. This value is: **0b01000001** Arm Limited.	0x41
[23:20]	Variant	Variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product. **0b0001** r1p2	0b0001
[19:16]	Architecture	Indicates the architecture code. This value is: **0b1111** Architecture is defined by ID registers	0b1111
[15:4]	PartNum	Primary Part Number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. **0b110101001110** MakaluELP	0xD4E
[3:0]	Revision	Revision number for the device. **0b0010** r1p2	0b0010

Accessibility

Component	Offset	Instance	Range
Debug	0xD00	MIDR_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.5.34 EDPFR, External Debug Processor Feature Register

Provides information about implemented PE features.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offsets (2)

0xD20,0xD24

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx 0001 xxxx 0001 0001 xxxx 0011 0001 0001 0001 0001 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-177: ext_edpfr bit assignments

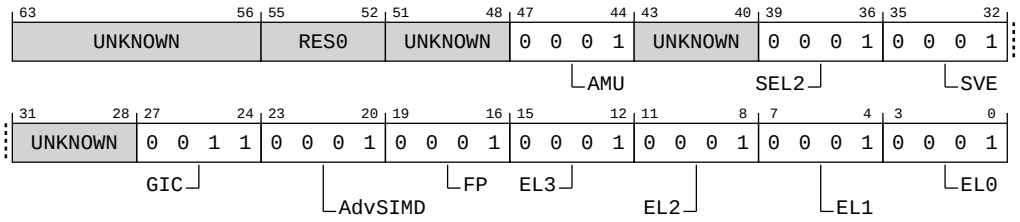


Table B-308: EDPFR bit descriptions

Bits	Name	Description	Reset
[63:56]	UNKNOWN	Reserved	UNKNOWN

Bits	Name	Description	Reset
[55:52]	RES0	Reserved	RES0
[51:48]	UNKNOWN	Reserved	UNKNOWN
[47:44]	AMU	Activity Monitors Extension. This value is : **0b0001** FEAT_AMUv1 is implemented.	0b0001
[43:40]	UNKNOWN	Reserved	UNKNOWN
[39:36]	SEL2	Secure EL2. This value is : **0b0001** Secure EL2 is implemented.	0b0001
[35:32]	SVE	Scalable Vector Extension. This value is : **0b0001** SVE is implemented.	0b0001
[31:28]	UNKNOWN	Reserved	UNKNOWN
[27:24]	GIC	System register GIC interface support. This field reads as 0x0 when GIC is disabled. **0b0011** System register interface to version 4.1 of the GIC CPU interface is supported.	0b0011
[23:20]	AdvSIMD	Advanced SIMD. This value is: **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	0b0001
[19:16]	FP	Floating Point. This value is: **0b0001** As for 0b0000, and also includes support for half-precision floating-point arithmetic.	0b0001
[15:12]	EL3	AArch64 EL3 Exception level handling **0b0001** EL3 can be executed in AArch64 state only.	0b0001
[11:8]	EL2	AArch64 EL2 Exception level handling **0b0001** EL2 can be executed in AArch64 state only.	0b0001
[7:4]	EL1	AArch64 EL1 Exception level handling **0b0001** EL1 can be executed in AArch64 state only.	0b0001
[3:0]	EL0	AArch64 EL0 Exception level handling **0b0001** EL0 can be executed in AArch64 state only.	0b0001

Accessibility

Component	Offset	Instance	Range
Debug	0xD20	EDPFR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

Component	Offset	Instance	Range
Debug	0xD24	EDPFR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

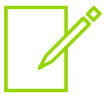
RO

Otherwise

ImplementationDefined

B.5.35 EDDFR, External Debug Feature Register

Provides top level information about the debug system.



Debuggers must use ext-EDDEVARCH to determine the Debug architecture version.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offsets (2)

0xD28,0xD2C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx 0001 xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0110 0001 xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-178: ext_eddfr bit assignments

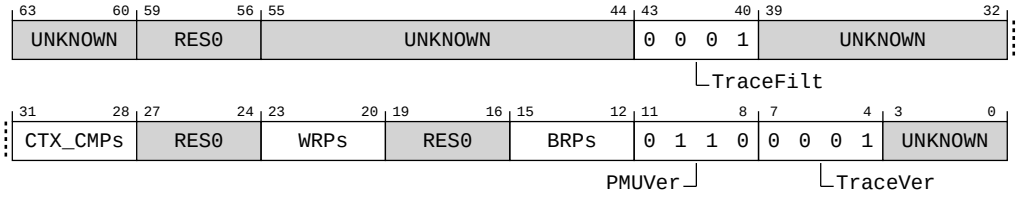


Table B-311: EDDFR bit descriptions

Bits	Name	Description	Reset
[63:60]	**UNKNOWN**	Reserved	**UNKNOWN**
[59:56]	**RES0**	Reserved	**RES0**
[55:44]	**UNKNOWN**	Reserved	**UNKNOWN**
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. This value is : **0b0001** Armv8.4 Self-hosted Trace Extension is implemented.	0b0001
[39:32]	**UNKNOWN**	Reserved	**UNKNOWN**
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. In an Armv8-A implementation that supports AArch64, this field returns the value of AArch64-ID_AA64DFR0_EL1.CTX_CMPs.	xxxx
[27:24]	**RES0**	Reserved	**RES0**
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0b0000 is reserved. In an Armv8-A implementation that supports AArch64, this field returns the value of AArch64-ID_AA64DFR0_EL1.WRPs.	xxxx
[19:16]	**RES0**	Reserved	**RES0**
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0b0000 is reserved. In an Armv8-A implementation that supports AArch64, this field returns the value of AArch64-ID_AA64DFR0_EL1.BRPs.	xxxx
[11:8]	PMUVer	Performance Monitors Extension version. **0b0110** PMUv3 for Armv8.5. As 0b0101, and adds support for: • 64-bit event counters. • If EL2 is implemented, the AArch64-MDCR_EL2.HCCD control. • If EL3 is implemented, the AArch64-MDCR_EL3.SCCD control.	0b0110

Bits	Name	Description	Reset
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE trace unit is implemented. **0b0001** Trace unit System registers implemented.	0b0001
[3:0]	UNKNOWN	Reserved	UNKNOWN

Accessibility

Component	Offset	Instance	Range
Debug	0xD28	EDDFR	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

Component	Offset	Instance	Range
Debug	0xD2C	EDDFR	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.5.36 EDAA32PFR, External Debug Auxiliary Processor Feature Register

Provides information about implemented PE features.


Note

The register mnemonic, EDAA32PFR, is derived from previous definitions of this register that defined this register only when AArch64 was not supported.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD60

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-179: ext_edaa32pfr bit assignments

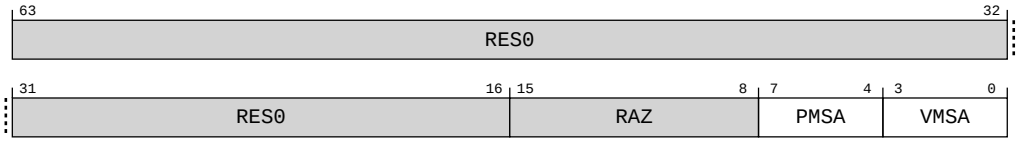


Table B-314: EDAA32PFR bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:8]	**RAZ**	Reserved	**RAZ**
[7:4]	PMSA	Indicates support for a 32-bit PMSA. Defined values are: **0b0000** PMSA-32 not supported. All other values are reserved. In Armv8-A, the only permitted value is `0b0000`.	xxxx

Bits	Name	Description	Reset
[3:0]	VMSA	Defined values are: **0b0000** VMSAv8-64 supported. All other values are reserved. In Armv8-A, the only permitted value is 0b0000.	xxxx

Accessibility

Component	Offset	Instance	Range
Debug	0xD60	EDAA32PFR	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.5.37 EDITCTRL, External Debug Integration mode Control register

Enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xF00

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-180: ext_editctrl bit assignments

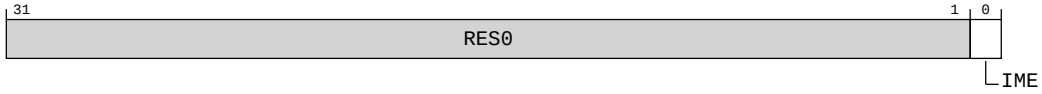


Table B-316: EDITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	IME	Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is **IMPLEMENTATION DEFINED**. **0b0** Normal operation. **0b1** Integration mode enabled.	x [6]

Accessibility

Component	Offset	Instance	Range
Debug	0xF00	EDITCTRL	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()
RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()
RW

Otherwise
ImplementationDefined

B.5.38 DBGCLAIMSET_EL1, Debug CLAIM Tag Set register

Used by software to set the CLAIM tag bits to 1.

The architecture does not define any functionality for the CLAIM tag bits.

[6] The following resets apply: - Whichever power domain the register is implemented in, this field resets to 0. - Otherwise, the value of this field is unchanged.

Note

CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the ext-DBGCLAIMCLR_EL1 register.

Configurations

An implementation must include eight CLAIM tag bits.

Attributes

Width

32

Component

Debug

Register offset

0xFA0

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-181: ext_dbgclaimset_el1 bit assignments

Table B-318: DBGCLAIMSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RAZ/WI	Reserved	RAZ/WI

Bits	Name	Description	Reset
[7:0]	CLAIM	Set CLAIM tag bits. This field is **RAO**. Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1. Writing 0 to one of these bits has no effect.	8{x}

Accessibility

Component	Offset	Instance	Range
Debug	0xFA0	DBGCLAIMSET_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()
RO

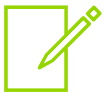
When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()
RW

Otherwise
ERROR

B.5.39 DBGCLAIMCLR_EL1, Debug CLAIM Tag Clear register

Used by software to read the values of the CLAIM tag bits, and to clear CLAIM tag bits to 0.

The architecture does not define any functionality for the CLAIM tag bits.


Note

CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the ext-DBGCLAIMSET_EL1 register.

Configurations

An implementation must include eight CLAIM tag bits.

Attributes

Width
32

Component
Debug

Register offset

0xFA4

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-182: ext_dbgclaimclr_el1 bit assignments

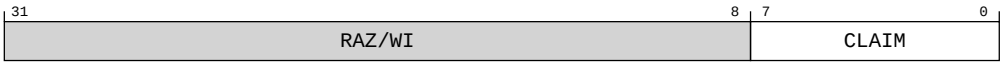


Table B-320: DBGCLAIMCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RAZ/WI	Reserved	RAZ/WI
[7:0]	CLAIM	Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits. Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0. Writing 0 to one of these bits has no effect.	0x00

Accessibility

Component	Offset	Instance	Range
Debug	0xFA4	DBGCLAIMCLR_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && SoftwareLockStatus()

RO

When IsCorePowered() && !DoubleLockStatus() && !OSLockStatus() && !SoftwareLockStatus()

RW

Otherwise

ERROR

B.5.40 EDDEVAFF0, External Debug Device Affinity register 0

Copy of the low half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFA8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-183: ext_eddevaff0 bit assignments

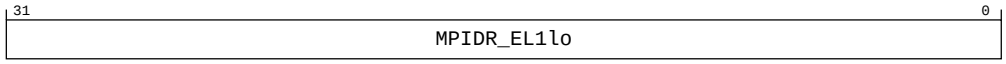


Table B-322: EDDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1lo	AArch64-MPIDR_EL1 low half. Read-only copy of the low half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32{x}

Accessibility

Component	Offset	Instance	Range
Debug	0xFA8	EDDEVAFF0	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.5.41 EDDEVAFF1, External Debug Device Affinity register 1

Copy of the high half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width
32

Component
Debug

Register offset
0xFAC

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-184: ext_eddevaff1 bit assignments

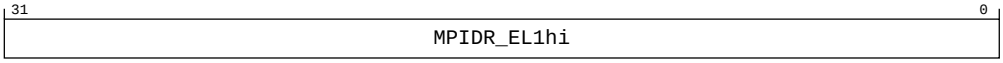


Table B-324: EDDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1hi	AArch64-MPIDR_EL1 high half. Read-only copy of the high half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	`32{x}`

Accessibility

Component	Offset	Instance	Range
Debug	0xFAC	EDDEVAFF1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.42 EDLAR, External Debug Lock Access Register

Allows or disallows access to the external debug registers through a memory-mapped interface.

The optional Software Lock provides a lock to prevent memory-mapped writes to the debug registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the debug registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

If FEAT_DoPD is implemented, Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Software uses EDLAR to set or clear the lock, and ext-EDLSR to check the current status of the lock.

Attributes

Width

32

Component

Debug

Register offset

0xFB0

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-185: ext_edlar bit assignments

Table B-326: EDLAR bit descriptions

Bits	Name	Description	Reset
[31:0]	KEY	Lock Access control. Writing the key value `0xC5ACCE55` to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface. Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.	`32{x}`

Accessibility

Component	Offset	Instance	Range
Debug	0xFB0	EDLAR	None

This interface is accessible as follows:

When IsCorePowered()

WO

Otherwise

ERROR

B.5.43 EDLSR, External Debug Lock Status Register

Indicates the current status of the software lock for external debug registers.

The optional Software Lock provides a lock to prevent memory-mapped writes to the debug registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the debug registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

If FEAT_DoPD is implemented, Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Software uses ext-EDLAR to set or clear the lock, and EDLSR to check the current status of the lock.

Attributes

Width

32

Component

Debug

Register offset

0xFB4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx1x

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-186: ext_edlsr bit assignments

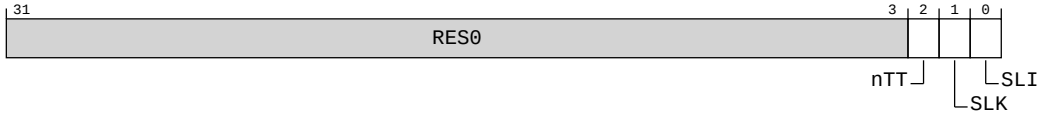


Table B-328: EDLSR bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2]	nTT	Not thirty-two bit access required. **RAZ**.	x
[1]	SLK	Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when Software Lock is not implemented, this field is **RES0**. For memory-mapped accesses when Software Lock is implemented, possible values of this field are: **0b0** Lock clear. Writes are permitted to this component's registers. **0b1** Lock set. Writes to this component's registers are ignored, and reads have no side effects.	0b1

Bits	Name	Description	Reset
[0]	SLI	Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is **IMPLEMENTATION DEFINED**. Permitted values are: **0b0** Software Lock not implemented or not memory-mapped access. **0b1** Software Lock implemented and memory-mapped access.	x

Accessibility

Component	Offset	Instance	Range
Debug	0xFB4	EDLSR	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.5.44 DBGAUTHSTATUS_EL1, Debug Authentication Status register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for debug.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width
32

Component
Debug

Register offset
0xFB8

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-187: ext_dbgauthstatus_el1 bit assignments

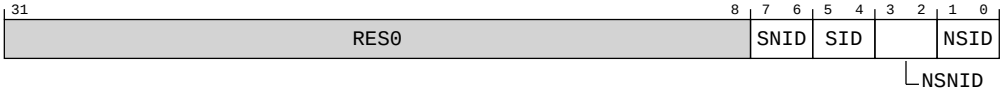


Table B-330: DBGAUTHSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:6]	SNID	Secure non-invasive debug. This field has the same value as DBGAUTHSTATUS_EL1.SID.	xx
[5:4]	SID	Secure invasive debug. **0b00** Not implemented. EL3 is not implemented and the Effective value of AArch64-SCR_EL3.NS is 1. **0b10** Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. **0b11** Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE. All other values are reserved.	xx
[3:2]	NSNID	Non-secure non-invasive debug. **0b00** Not implemented. EL3 is not implemented and the Effective value of AArch64-SCR_EL3.NS is 0. **0b11** Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE. All other values are reserved.	xx
[1:0]	NSID	Non-secure invasive debug. **0b00** Not implemented. EL3 is not implemented and the Effective value of AArch64-SCR_EL3.NS is 0. **0b10** Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE. **0b11** Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE. All other values are reserved.	xx

Accessibility

Component	Offset	Instance	Range
Debug	0xFB8	DBGAUTHSTATUS_EL1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.45 EDDEVARCH, External Debug Device Architecture register

Identifies the programmers' model architecture of the external debug component.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFBC

Access type

See bit descriptions

Reset value

0100 0111 0111 0000 1001 1010 0001 0101

Bit descriptions

Figure B-188: ext_eddevarch bit assignments

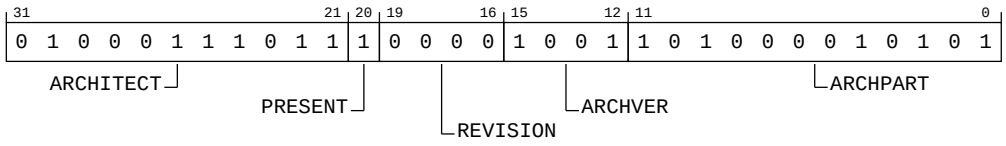


Table B-332: EDDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For debug, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, `0x4`. Bits [27:21] are the JEP106 ID code, `0x3B`. **0b01000111011**	`0b01000111011`
[20]	PRESENT	Indicates that the DEVARCH is present. **0b1**	`0b1`
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For debug, the revision defined by Armv8 is `0x0`. All other values are reserved. **0b0000**	`0b0000`
[15:12]	ARCHVER	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, FEAT_Debugv8p4.	`0b1001`
[11:0]	ARCHPART	The fields ARCHVER and ARCHPART together form the field ARCHID, so that ARCHPART is ARCHID[11:0]. **0b101000010101** Armv8-A debug architecture.	`0xA15`

Accessibility

Component	Offset	Instance	Range
Debug	0xFBC	EDDEVARCH	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.46 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug implementation.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC0

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-189: ext_eddevid2 bit assignments

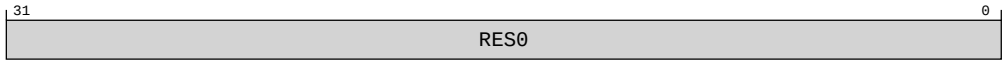


Table B-334: EDDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
Debug	0xFC0	EDDEVID2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.47 EDDEVID1, External Debug Device ID register 1

Provides extra information for external debuggers about features of the debug implementation.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-190: ext_eddevid1 bit assignments

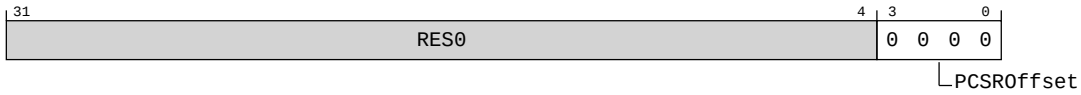


Table B-336: EDDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	PCSROffset	This field indicates the offset applied to PC samples returned by reads of ext-EDPCSR. **0b0000** ext-EDPCSR not implemented.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC4	EDDEVID1	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.5.48 EDDEVID, External Debug Device ID register 0

Provides extra information for external debuggers about features of the debug implementation.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain.

If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width
32

Component
Debug

Register offset
0xFC8

Access type
See bit descriptions

Reset value
xxxx 0000 xxxx xxxx xxxx xxxx 0001 0000


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-191: ext_eddevid bit assignments

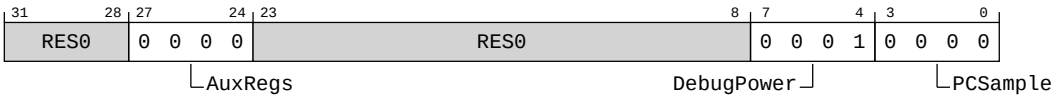


Table B-338: EDDEVID bit descriptions

Bits	Name	Description	Reset
[31:28]	**RES0**	Reserved	**RES0**
[27:24]	AuxRegs	Indicates support for Auxiliary registers. **0b0000** None supported.	0b0000
[23:8]	**RES0**	Reserved	**RES0**
[7:4]	DebugPower	Indicates support for the ARMv8.3-DoPD feature. **0b0001** FEAT_DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain.	0b0001
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using external debug registers. **0b0000** PC Sample-based Profiling Extension is not implemented in the external debug registers space.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC8	EDDEVID	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.49 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PEs debug logic.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFCC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-192: ext_eddevtype bit assignments

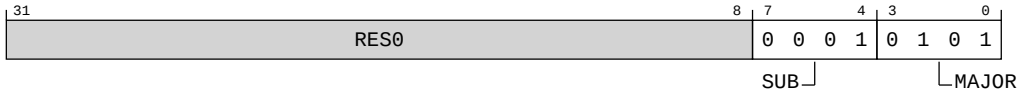


Table B-340: EDDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SUB	Subtype. Indicates this is a component within a PE. **0b0001**	0b0001
[3:0]	MAJOR	Major type. Indicates this is a debug logic component. **0b0101**	0b0101

Accessibility

Component	Offset	Instance	Range
Debug	0xFCC	EDDEVTYPE	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.50 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFD0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-193: ext_edpidr4 bit assignments

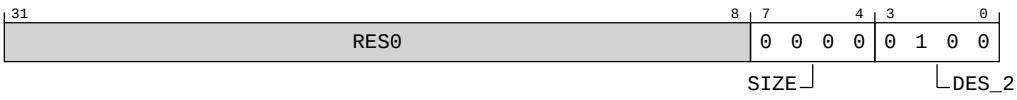


Table B-342: EDPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	4KB count. **0b0000**	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited. This is bits[3:0] of the JEP106 continuation code.	0b0100

Accessibility

Component	Offset	Instance	Range
Debug	0xFD0	EDPIDR4	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.51 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0100 1110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-194: ext_edpidr0 bit assignments

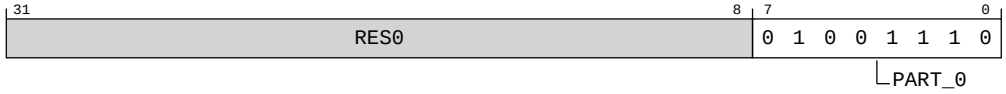


Table B-344: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b01001110** Least Significant byte of the debug part number	0x4E

Accessibility

Component	Offset	Instance	Range
Debug	0xFE0	EDPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.52 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-195: ext_edpidr1 bit assignments

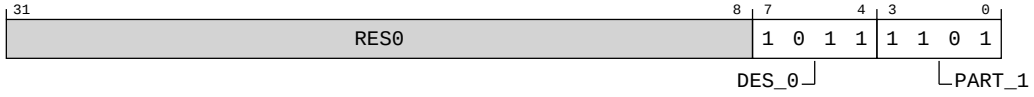


Table B-346: EDPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Arm Limited. This is the least significant nibble of JEP106 ID code.	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** Part number, most significant nibble.	`0b1101`

Accessibility

Component	Offset	Instance	Range
Debug	0xFE4	EDPIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.53 EDPIDR2, External Debug Peripheral Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-196: ext_edpidr2 bit assignments

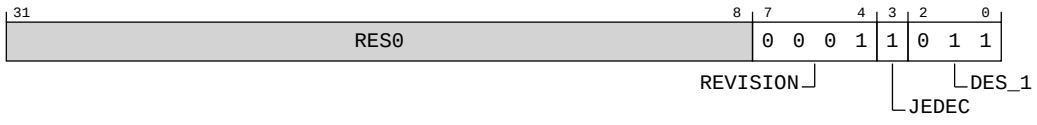


Table B-348: EDPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited. This is bits[6:4] of the JEP106 ID code.	0b011

Accessibility

Component	Offset	Instance	Range
Debug	0xFE8	EDPIDR2	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.5.54 EDPIDR3, External Debug Peripheral Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width
32

Component
Debug

Register offset
0xFEC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-197: ext_edpidr3 bit assignments

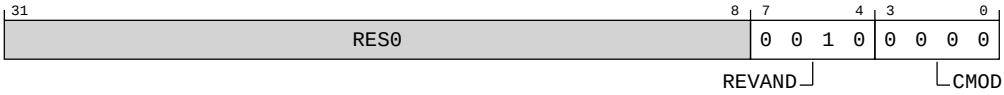


Table B-350: EDPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010**	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000** The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFEC	EDPIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.55 EDCIDR0, External Debug Component Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-198: ext_edcidr0 bit assignments

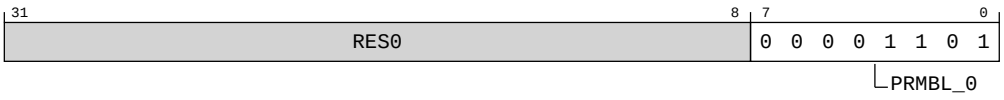


Table B-352: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
Debug	0xFF0	EDCIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.56 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-199: ext_edcidr1 bit assignments

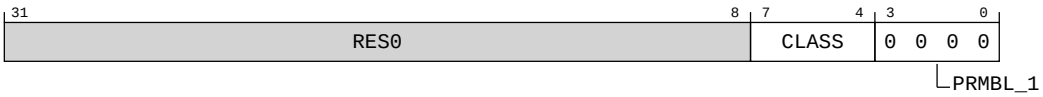


Table B-354: EDCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component. Other values are defined by the CoreSight Architecture. This field reads as 0x9.	xxxx
[3:0]	PRMBL_1	Preamble. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFF4	EDCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.57 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-200: ext_edcidr2 bit assignments

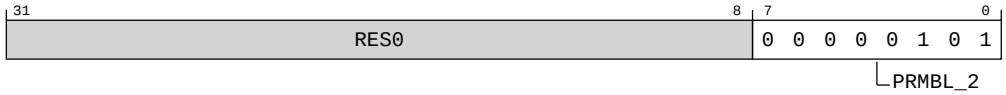


Table B-356: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0b00000101**	0x05

Accessibility

Component	Offset	Instance	Range
Debug	0xFF8	EDCIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5.58 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFFC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-201: ext_edcidr3 bit assignments

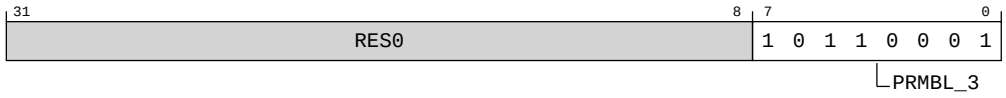


Table B-358: EDCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PRMBL_3	Preamble. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
Debug	0xFFC	EDCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.6 External AMU registers summary

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

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

Table B-360: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00 [31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0x4	AMEVCNTR00 [63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x8	AMEVCNTR01 [31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0xC	AMEVCNTR01 [63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02 [31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0x14	AMEVCNTR02 [63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03 [31:0]	–	32-bit	Activity Monitors Event Counter Registers 0
0x1C	AMEVCNTR03 [63:32]	–	32-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10 [31:0]	–	32-bit	Activity Monitors Event Counter Registers 1
0x104	AMEVCNTR10 [63:32]	–	32-bit	Activity Monitors Event Counter Registers 1
0x108	AMEVCNTR11 [31:0]	–	32-bit	Activity Monitors Event Counter Registers 1
0x10C	AMEVCNTR11 [63:32]	–	32-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12 [31:0]	–	32-bit	Activity Monitors Event Counter Registers 1
0x114	AMEVCNTR12 [63:32]	–	32-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	–	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	–	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	–	32-bit	Activity Monitors Event Type Registers 0

Offset	Name	Reset	Width	Description
0x40C	AMEVTYPER03	—	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	—	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	—	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	—	32-bit	Activity Monitors Event Type Registers 1
0x48C	AMEVTYPER13	—	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	—	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	—	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	—	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	—	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	—	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	—	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	—	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	—	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	—	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	—	32-bit	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	—	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	—	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	—	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	—	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	—	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	—	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	—	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	—	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	—	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	—	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	—	32-bit	Activity Monitors Component Identification Register 3

B.6.1 AMEVCNTR00, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offsets (2)

0x0,0x4

Access type

Read

R

Write

RESERVED

Reset value

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

Bit descriptions

Figure B-202: ext_amevcntr00 bit assignments

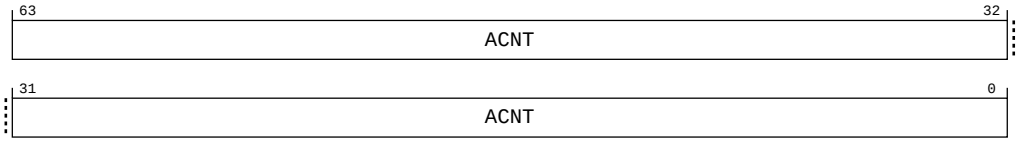


Table B-361: AMEVCNTR00 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Architected activity monitor event counter n. Value of architected activity monitor event counter n, where n is the number of this register and is a number from 0 to 3.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.


Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x0	AMEVCNTR00	31:0

This interface is accessible as follows:

RO

Component	Offset	Instance	Range
AMU	0x4	AMEVCNTR00	63:32

This interface is accessible as follows:

RO

B.6.2 AMEVCNTR01, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offsets (2)

0x8,0xC

Access type

Read

R

Write

RESERVED

Reset value

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

Bit descriptions

Figure B-203: ext_amevcntr01 bit assignments

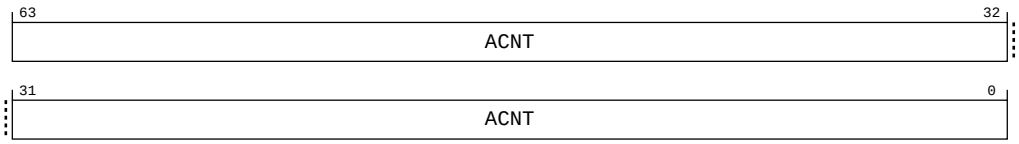


Table B-364: AMEVCNTR01 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Architected activity monitor event counter n. Value of architected activity monitor event counter n, where n is the number of this register and is a number from 0 to 3.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x8	AMEVCNTR01	31:0

This interface is accessible as follows:

RO

Component	Offset	Instance	Range
AMU	0xC	AMEVCNTR01	63:32

This interface is accessible as follows:

RO

B.6.3 AMEVCNTR02, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offsets (2)

0x10,0x14

Access type

Read

R

Write

RESERVED

Reset value

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

Bit descriptions

Figure B-204: ext_amevcntr02 bit assignments

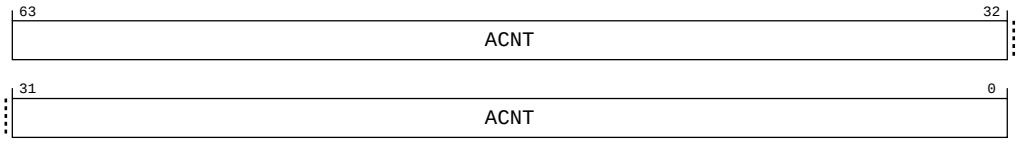


Table B-367: AMEVCNTR02 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Architected activity monitor event counter n. Value of architected activity monitor event counter n, where n is the number of this register and is a number from 0 to 3.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x10	AMEVCNTR02	31:0

This interface is accessible as follows:

RO

Component	Offset	Instance	Range
AMU	0x14	AMEVCNTR02	63:32

This interface is accessible as follows:

RO

B.6.4 AMEVCNTR03, Activity Monitors Event Counter Registers 0

Provides access to the architected activity monitor event counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offsets (2)

0x18,0x1C

Access type

Read

R

Write

RESERVED

Reset value

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

Bit descriptions

Figure B-205: ext_amevcntr03 bit assignments

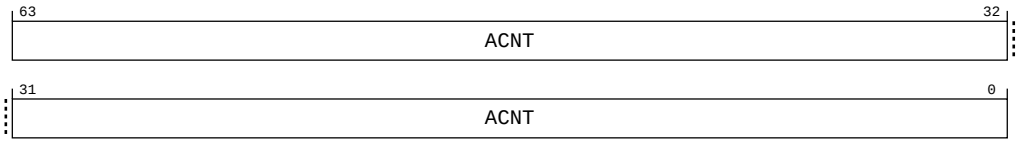


Table B-370: AMEVCNTR03 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Architected activity monitor event counter n. Value of architected activity monitor event counter n, where n is the number of this register and is a number from 0 to 3.	0x0000000000000000

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access*

requirements for reserved and unallocated registers in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVCNTR0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x18	AMEVCNTR03	31:0

This interface is accessible as follows:

RO

Component	Offset	Instance	Range
AMU	0x1C	AMEVCNTR03	63:32

This interface is accessible as follows:

RO

B.6.5 AMEVCNTR10, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offsets (2)

0x100,0x104

Access type

Read

R

Write

RESERVED

Reset value

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

Bit descriptions

Figure B-206: ext_amevcntr10 bit assignments

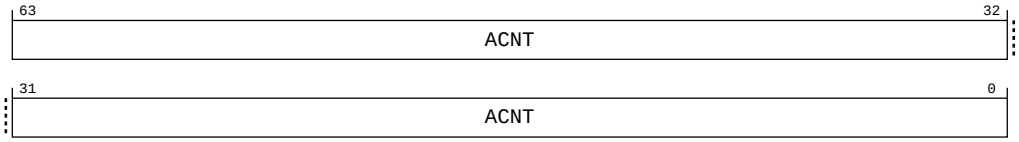


Table B-373: AMEVCNTR10 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Auxiliary activity monitor event counter n. Value of auxiliary activity monitor event counter n, where n is the number of this register and is a number from 0 to 15.	0x0000000000000000

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR1<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.


Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x100	AMEVCNTR10	31:0

This interface is accessible as follows:

RO

Component	Offset	Instance	Range
AMU	0x104	AMEVCNTR10	63:32

This interface is accessible as follows:

RO

B.6.6 AMEVCNTR11, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offsets (2)

0x108,0x10C

Access type

Read

R

Write

RESERVED

Reset value

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

Bit descriptions

Figure B-207: ext_amevcntr11 bit assignments

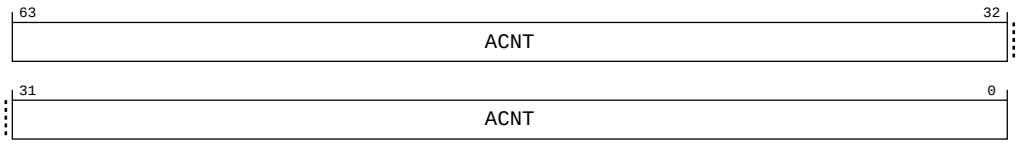


Table B-376: AMEVCNTR11 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Auxiliary activity monitor event counter n. Value of auxiliary activity monitor event counter n, where n is the number of this register and is a number from 0 to 15.	0x0000000000000000

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR1<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x108	AMEVCNTR11	31:0

This interface is accessible as follows:

RO

Component	Offset	Instance	Range
AMU	0x10C	AMEVCNTR11	63:32

This interface is accessible as follows:

RO

B.6.7 AMEVCNTR12, Activity Monitors Event Counter Registers 1

Provides access to the auxiliary activity monitor event counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

AMU

Register offsets (2)

0x110,0x114

Access type

Read

R

Write

RESERVED

Reset value

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

Bit descriptions

Figure B-208: ext_amevcntr12 bit assignments

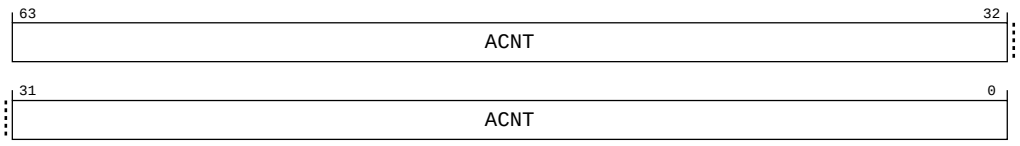


Table B-379: AMEVCNTR12 bit descriptions

Bits	Name	Description	Reset
[63:0]	ACNT	Auxiliary activity monitor event counter n. Value of auxiliary activity monitor event counter n, where n is the number of this register and is a number from 0 to 15.	0x0000000000000000

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR1<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVCNTR1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x110	AMEVCNTR12	31:0

This interface is accessible as follows:

RO

Component	Offset	Instance	Range
AMU	0x114	AMEVCNTR12	63:32

This interface is accessible as follows:

RO

B.6.8 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

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0001 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-209: ext_amevtyper00 bit assignments

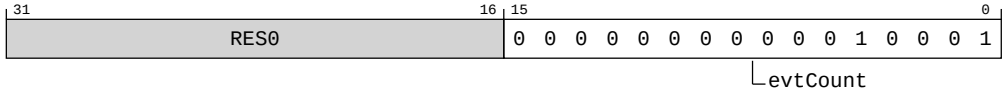


Table B-382: AMEVTYPER00 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000010001** Processor frequency cycles	0x0011

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x400	AMEVTYPER00	None

This interface is accessible as follows:

RO

B.6.9 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

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx 0100 0000 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-210: ext_amevtyper01 bit assignments

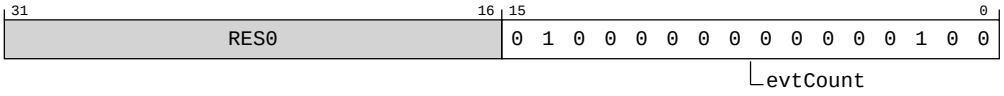


Table B-384: AMEVTYPER01 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000100** Constant frequency cycles	0x4004

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access*

requirements for reserved and unallocated registers in the *Arm® Architecture Reference Manual for A-profile architecture*.


Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.


Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x404	AMEVTYPER01	None

This interface is accessible as follows:

RO

B.6.10 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

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 1000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-211: ext_amevtyper02 bit assignments

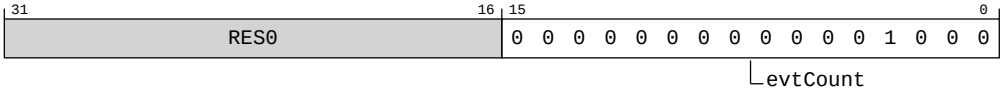


Table B-386: AMEVTYPER02 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000001000** Instructions retired	0x0008

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.


Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x408	AMEVTYPER02	None

This interface is accessible as follows:

RO

B.6.11 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

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx 0100 0000 0000 0101


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-212: ext_amevtyper03 bit assignments

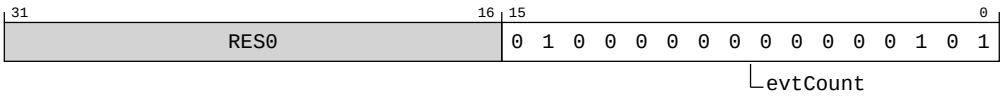


Table B-388: AMEVTYPER03 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0100000000000101** Memory stall cycles	0x4005

Access

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads of AMEVTYPER0<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x40C	AMEVTYPER03	None

This interface is accessible as follows:

RO

B.6.12 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

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-213: ext_amevtyper10 bit assignments

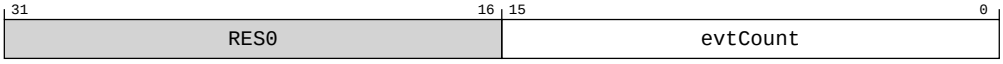


Table B-390: AMEVTYPER10 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is **IMPLEMENTATION DEFINED** what values are supported by each counter.	`16{x}`

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x480	AMEVTYPER10	None

This interface is accessible as follows:

RO

B.6.13 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

Access type

Read

R

Write

RESERVED

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-214: ext_amevtyper11 bit assignments

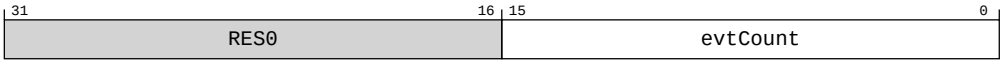


Table B-392: AMEVTYPER11 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is **IMPLEMENTATION DEFINED** what values are supported by each counter.	`16{x}`

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.


Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.


Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x484	AMEVTYPER11	None

This interface is accessible as follows:

RO

B.6.14 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

Access type

Read

R

Write

RESERVED

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-215: ext_amevtyper12 bit assignments

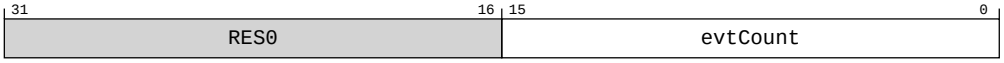


Table B-394: AMEVTYPER12 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is **IMPLEMENTATION DEFINED** what values are supported by each counter.	`16{x}`

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x488	AMEVTYPER12	None

This interface is accessible as follows:

RO

B.6.15 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

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-216: ext_amevtyper13 bit assignments

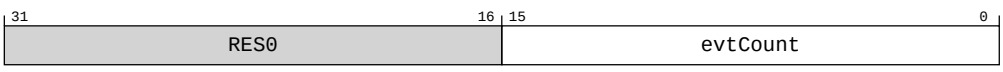


Table B-396: AMEVTYPER13 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1<n>. It is **IMPLEMENTATION DEFINED** what values are supported by each counter.	`16{x}`

Access

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Accessibility

If <n> is greater than or equal to the number of auxiliary activity monitor event counters, reads of AMEVTYPER1<n> are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

ext-AMCGCR.CG1NC identifies the number of auxiliary activity monitor event counters.

Component	Offset	Instance	Range
AMU	0x48C	AMEVTYPER13	None

This interface is accessible as follows:

RO

B.6.16 AMCNTENSET0, Activity Monitors Count Enable Set Register 0

Enable control bits for the architected activity monitors event counters, ext-AMEVCNTR0<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xC00

Access type

RO

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-217: ext_amcntenset0 bit assignments

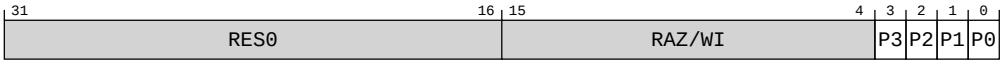


Table B-398: AMCNTENSET0 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:4]	**RAZ/WI**	Reserved	**RAZ/WI**
[3:0]	P<n>, bit[n], where n = 3 to 0	Activity monitor event counter enable bit for ext-AMEVCNTR0<n>. **Note:** ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that ext-AMEVCNTR0<n> is disabled. **0b1** When read, means that ext-AMEVCNTR0<n> is enabled.	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xC00	AMCNTENSET0	None

This interface is accessible as follows:

RO

B.6.17 AMCNTENSET1, Activity Monitors Count Enable Set Register 1

Enable control bits for the auxiliary activity monitors event counters, ext-AMEVCNTR1<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xC04

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-218: ext_amcntenset1 bit assignments

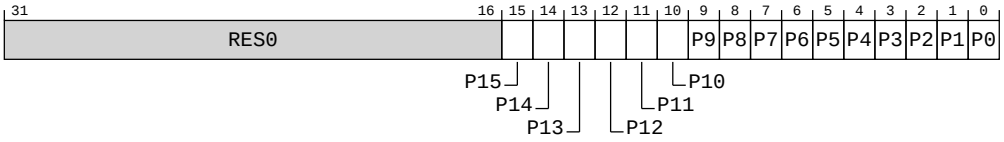


Table B-400: AMCNTENSET1 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	P<n>, bit[n], where n = 15 to 0	Activity monitor event counter enable bit for ext-AMEVCNTR1<n>. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in ext-AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that ext-AMEVCNTR1<n> is disabled. **0b1** When read, means that ext-AMEVCNTR1<n> is enabled.	0x0000

Access

If the number of auxiliary activity monitor event counters implemented is zero, reads of AMCNTENSET1 are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

The number of auxiliary activity monitor counters implemented is zero exactly when ext-AMCFGR.NCG == 0b0000.

Accessibility

If the number of auxiliary activity monitor event counters implemented is zero, reads of AMCNTENSET1 are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

Note

The number of auxiliary activity monitor counters implemented is zero exactly when ext-AMCFGR.NCG == 0b0000.

Component	Offset	Instance	Range
AMU	0xC04	AMCNTENSET1	None

This interface is accessible as follows:

RO

B.6.18 AMCNTENCLR0, Activity Monitors Count Enable Clear Register 0

Disable control bits for the architected activity monitors event counters, ext-AMEVCNTR0<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xC20

Access type

RO

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-219: ext_amcntenclr0 bit assignments

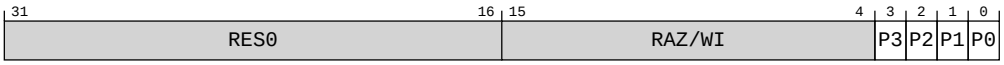


Table B-402: AMCNTENCLR0 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:4]	**RAZ/WI**	Reserved	**RAZ/WI**

Bits	Name	Description	Reset
[3:0]	P<n>, bit[n], where n = 3 to 0	Activity monitor event counter disable bit for ext-AMEVCNTR0<n>. **Note:** ext-AMCGCR.CG0NC identifies the number of architected activity monitor event counters. In an implementation that includes FEAT_AMUv1, the number of architected activity monitor event counters is 4. Possible values of each bit are: **0b0** When read, means that ext-AMEVCNTR0<n> is disabled. **0b1** When read, means that ext-AMEVCNTR0<n> is enabled.	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xC20	AMCNTENCLR0	None

This interface is accessible as follows:

RO

B.6.19 AMCNTENCLR1, Activity Monitors Count Enable Clear Register 1

Disable control bits for the auxiliary activity monitors event counters, ext-AMEVCNTR1<n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xC24

Access type

Read

R

Write

RESERVED

Reset value

xxxx xxxx xxxx xxxx 0000 0000 0000 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-220: ext_amcntenclr1 bit assignments

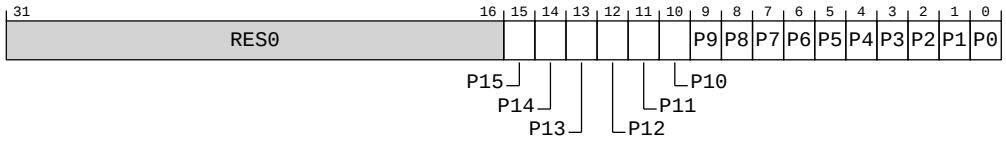


Table B-404: AMCNTENCLR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	P<n>, bit[n], where n = 15 to 0	Activity monitor event counter disable bit for ext-AMEVCNTR1<n>. When N is less than 16, bits [15:N] are **RAZ**, where N is the value in ext-AMCGCR.CG1NC. Possible values of each bit are: **0b0** When read, means that ext-AMEVCNTR1<n> is disabled. **0b1** When read, means that ext-AMEVCNTR1<n> is enabled.	0x0000

Access

If the number of auxiliary activity monitor event counters implemented is zero, reads of AMCNTENCLR1 are RAZ. Software must treat reserved accesses as **RES0**. See *Access requirements for reserved and unallocated registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

The number of auxiliary activity monitor event counters implemented is zero exactly when ext-AMCFGR.NCG == 0b0000.

Accessibility

If the number of auxiliary activity monitor event counters implemented is zero, reads of AMCNTENCLR1 are RAZ. Software must treat reserved accesses as RES0. See 'Access requirements for reserved and unallocated registers'.

The number of auxiliary activity monitor event counters implemented is zero exactly when ext-AMCFGR.NCG == 0b0000.

Component	Offset	Instance	Range
AMU	0xC24	AMCNTENCLR1	None

This interface is accessible as follows:

RO

B.6.20 AMCGCR, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xCE0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx 0000 0011 0000 0100

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-221: ext_amcgcr bit assignments

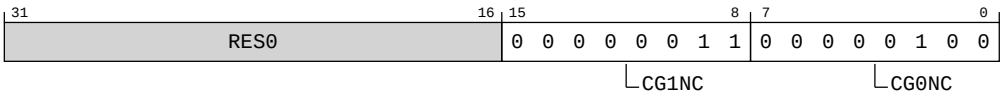


Table B-406: AMCGCR bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group. In an implementation that includes FEAT_AMUv1, the permitted range of values is 0 to 16. **0b00000011** Three counters in the auxiliary counter group	0x03
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group. **0b00000100**	0x04

Accessibility

Component	Offset	Instance	Range
AMU	0xCE0	AMCGCR	None

This interface is accessible as follows:

RO

B.6.21 AMCFGR, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR is applicable to both the architected and the auxiliary counter groups.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE00

Access type

RO

Reset value

0001 xxx1 0000 0000 0011 1111 0000 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-222: ext_amcfgr bit assignments

Table B-408: AMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. **0b0001** Two counter groups are implemented	0b0001
[27:25]	**RES0**	Reserved	**RES0**
[24]	HDBG	Halt-on-debug supported. This feature must be supported, and so this bit is 0b1. **0b1** ext-AMCR.HDBG is read/write.	0b1
[23:14]	**RAZ**	Reserved	**RAZ**
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the Activity Monitors Extension is [AMCFGR.SIZE + 1]. The counters are 64-bit. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. The counters are at doubleword-aligned addresses. **0b111111**	0b111111

Bits	Name	Description	Reset
[7:0]	N	Defines the number of activity monitor event counters. The total number of counters implemented in all groups by the Activity Monitors Extension is [AMCFGR.N + 1]. **0b00000110** Seven activity monitor event counters	0x06

Accessibility

Component	Offset	Instance	Range
AMU	0xE00	AMCFGR	None

This interface is accessible as follows:

RO

B.6.22 AMCR, Activity Monitors Control Register

Global control register for the activity monitors implementation. AMCR is applicable to both the architected and the auxiliary counter groups.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE04

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-223: ext_amcr bit assignments

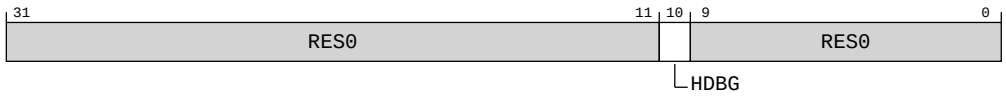


Table B-410: AMCR bit descriptions

Bits	Name	Description	Reset
[31:11]	**RES0**	Reserved	**RES0**
[10]	HDBG	This bit controls whether activity monitor counting is halted when the PE is halted in Debug state. **0b0** Activity monitors do not halt counting when the PE is halted in Debug state. **0b1** Activity monitors halt counting when the PE is halted in Debug state.	x
[9:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
AMU	0xE04	AMCR	None

This interface is accessible as follows:

RO

B.6.23 AMIIDR, Activity Monitors Implementation Identification Register

Defines the implementer and revisions of the AMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE08

Access type

RO

Reset value

1101 0100 1110 0001 0010 xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-224: ext_amiidr bit assignments

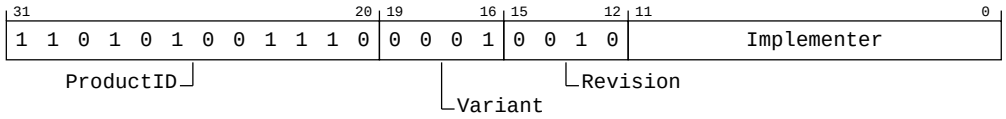


Table B-412: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. **0b110101001110** MakaluELP 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.	0xD4E
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. **0b0001** r1p2 If ext-AMPIDR2 is implemented, ext-AMPIDR2.REVISION matches AMIIDR.Variant.	0b0001
[15:12]	Revision	This field distinguishes minor revisions of the product. **0b0010** r1p2 If ext-AMPIDR3 is implemented, ext-AMPIDR3.REVAND matches AMIIDR.Revision.	0b0010

Bits	Name	Description	Reset
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the AMU. For an Arm implementation, this field reads as `0x43B`. 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.	`12{x}`

Accessibility

Component	Offset	Instance	Range
AMU	0xE08	AMIIDR	None

This interface is accessible as follows:

RO

B.6.24 AMDEVAFF0, Activity Monitors Device Affinity Register 0

Copy of the low half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the AMU component relates to.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFA8

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-225: ext_amdevaff0 bit assignments

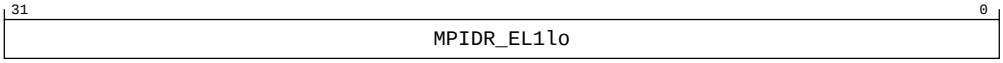


Table B-414: AMDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1lo	AArch64-MPIDR_EL1 low half. Read-only copy of the low half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32{x}

Accessibility

Component	Offset	Instance	Range
AMU	0xFA8	AMDEVAFF0	None

This interface is accessible as follows:

RO

B.6.25 AMDEVAFF1, Activity Monitors Device Affinity Register 1

Copy of the high half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the AMU component relates to.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFAC

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-226: ext_amdevaff1 bit assignments

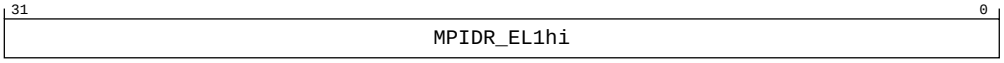


Table B-416: AMDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1hi	AArch64-MPIDR_EL1 high half. Read-only copy of the high half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32{x}

Accessibility

Component	Offset	Instance	Range
AMU	0xFAC	AMDEVAFF1	None

This interface is accessible as follows:

RO

B.6.26 AMDEVARCH, Activity Monitors Device Architecture Register

Identifies the programmers' model architecture of the AMU component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 0000 0000 1010 0110 0110

Bit descriptions

Figure B-227: ext_amdevarch bit assignments

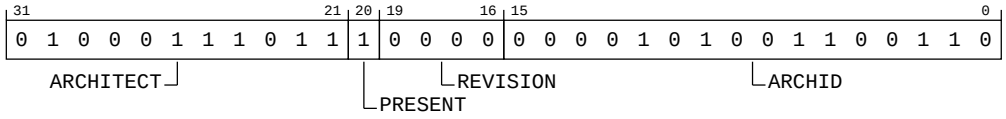


Table B-418: AMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For AMU, this is Arm Limited. **0b01000111011**	0b01000111011
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. **0b1**	0b1
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. **0b0000** Architecture revision is AMUv1. All other values are reserved.	0b0000
[15:0]	ARCHID	Defines this part to be an AMU component. For architectures defined by Arm this is further subdivided. For AMU: • Bits [15:12] are the architecture version, 0x0. • Bits [11:0] are the architecture part number, 0xA66. This corresponds to AMU architecture version AMUv1. **0b0000101001100110**	0x0A66

Accessibility

Component	Offset	Instance	Range
AMU	0xFBC	AMDEVARCH	None

This interface is accessible as follows:

RO

B.6.27 AMDEVTYPE, Activity Monitors Device Type Register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFCC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 0110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-228: ext_amdevtype bit assignments

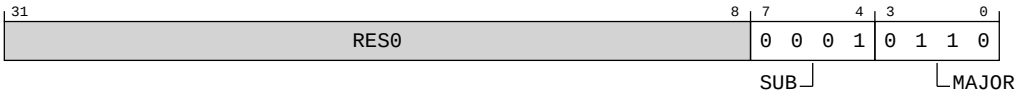


Table B-420: AMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Reads as `0x1`, to indicate this is a component within a PE. **`0b0001`** Component within a PE.	`0b0001`
[3:0]	MAJOR	Major type. Reads as `0x6`, to indicate this is a performance monitor component. **`0b0110`** Performance monitor component	`0b0110`

Accessibility

Component	Offset	Instance	Range
AMU	0xFCC	AMDEVTYPE	None

This interface is accessible as follows:

RO

B.6.28 AMPIDR4, Activity Monitors Peripheral Identification Register 4

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFD0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-229: ext_ampidr4 bit assignments

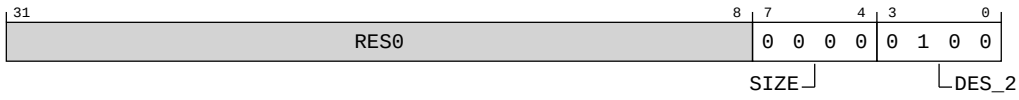


Table B-422: AMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	4KB count. **0b0000**	0b0000
[3:0]	DES_2	Designer. JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited. This is bits[3:0] of the JEP106 continuation code.	0b0100

Accessibility

Component	Offset	Instance	Range
AMU	0xFD0	AMPIDR4	None

This interface is accessible as follows:

RO

B.6.29 AMPIDR0, Activity Monitors Peripheral Identification Register 0

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0100 1110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-230: ext_ampidr0 bit assignments

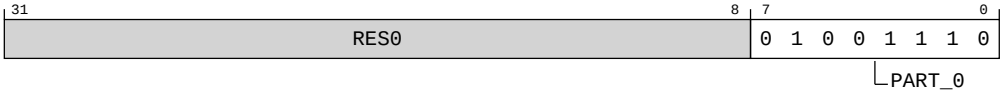


Table B-424: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b01001110** Part number, least significant byte.	0x4E

Accessibility

Component	Offset	Instance	Range
AMU	0xFE0	AMPIDR0	None

This interface is accessible as follows:

RO

B.6.30 AMPIDR1, Activity Monitors Peripheral Identification Register 1

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-231: ext_ampidr1 bit assignments

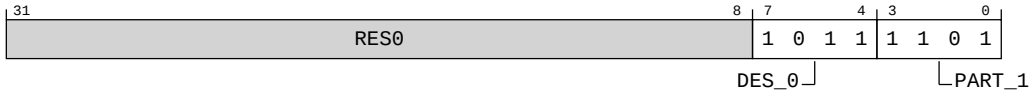


Table B-426: AMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is `0b1011`. **`0b1011`** Designer, least significant nibble of JEP106 ID code.	`0b1011`
[3:0]	PART_1	Part number, most significant nibble. **`0b1101`** Part number, most significant nibble.	`0b1101`

Accessibility

Component	Offset	Instance	Range
AMU	0xFE4	AMPIDR1	None

This interface is accessible as follows:

RO

B.6.31 AMPIDR2, Activity Monitors Peripheral Identification Register 2

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-232: ext_ampidr2 bit assignments

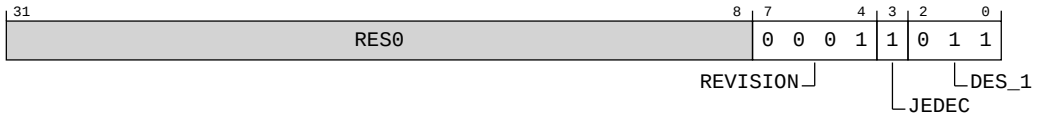


Table B-428: AMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0001** r1p2	0b0001

Bits	Name	Description	Reset
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited. This is bits[6:4] of the JEP106 ID code.	0b011

Accessibility

Component	Offset	Instance	Range
AMU	0xFE8	AMPIDR2	None

This interface is accessible as follows:

RO

B.6.32 AMPIDR3, Activity Monitors Peripheral Identification Register 3

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFEC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-233: ext_ampidr3 bit assignments

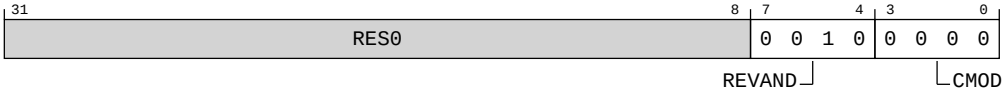


Table B-430: AMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using ext-AMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0010**	0b0010
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **0b0000** The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xFEC	AMPIDR3	None

This interface is accessible as follows:

RO

B.6.33 AMCIDR0, Activity Monitors Component Identification Register 0

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF0

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-234: ext_amcidr0 bit assignments

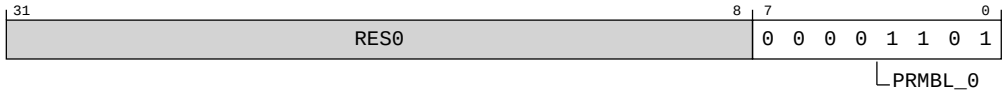


Table B-432: AMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
AMU	0xFF0	AMCIDR0	None

This interface is accessible as follows:

RO

B.6.34 AMCIDR1, Activity Monitors Component Identification Register 1

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF4

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-235: ext_amcidr1 bit assignments

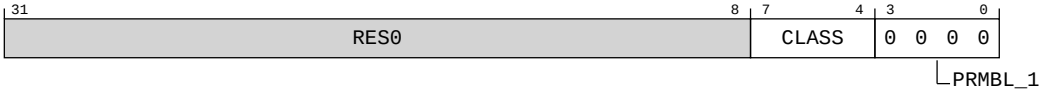


Table B-434: AMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight component.	xxxx
[3:0]	PRMBL_1	Preamble. **0b0000**	0b0000

Accessibility

Component	Offset	Instance	Range
AMU	0xFF4	AMCIDR1	None

This interface is accessible as follows:

RO

B.6.35 AMCIDR2, Activity Monitors Component Identification Register 2

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFF8

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-236: ext_amcidr2 bit assignments

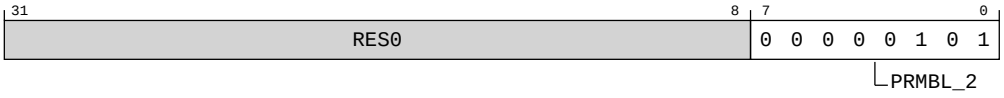


Table B-436: AMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Preamble. **0b00000101**	0x05

Accessibility

Component	Offset	Instance	Range
AMU	0xFF8	AMCIDR2	None

This interface is accessible as follows:

RO

B.6.36 AMCIDR3, Activity Monitors Component Identification Register 3

Provides information to identify an activity monitors component.

For more information, see *About the Component identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFFC

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-237: ext_amcidr3 bit assignments

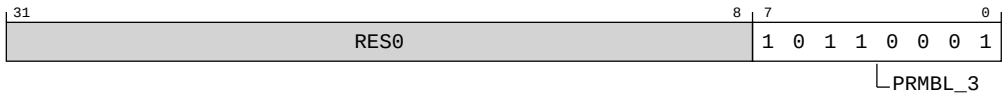


Table B-438: AMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
AMU	0xFFC	AMCIDR3	None

This interface is accessible as follows:

RO

B.7 External ETE registers summary

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

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

Table B-440: ETE registers summary

Offset	Name	Reset	Width	Description
0x004	TRCPRGCTLR	–	32-bit	Programming Control Register
0x00C	TRCSTATR	–	32-bit	Trace Status Register
0x010	TRCCONFIGR	–	32-bit	Trace Configuration Register
0x018	TRCAUXCTLR	–	32-bit	Auxiliary Control Register
0x020	TRCEVENTCTL0R	–	32-bit	Event Control 0 Register
0x024	TRCEVENTCTL1R	–	32-bit	Event Control 1 Register
0x028	TRCRSR	–	32-bit	Resources Status Register
0x030	TRCTSCTLR	–	32-bit	Timestamp Control Register
0x034	TRCSYNCPR	–	32-bit	Synchronization Period Register
0x038	TRCCCCTLR	–	32-bit	Cycle Count Control Register
0x03C	TRCBBCTLR	–	32-bit	Branch Broadcast Control Register
0x040	TRCTRACEIDR	–	32-bit	Trace ID Register
0x080	TRCVICTLR	–	32-bit	ViewInst Main Control Register
0x084	TRCVIIECTLR	–	32-bit	ViewInst Include/Exclude Control Register
0x088	TRCVISSCTLR	–	32-bit	ViewInst Start/Stop Control Register
0x100	TRCSEQEVR0	–	32-bit	Sequencer State Transition Control Register <n>
0x104	TRCSEQEVR1	–	32-bit	Sequencer State Transition Control Register <n>
0x108	TRCSEQEVR2	–	32-bit	Sequencer State Transition Control Register <n>

Offset	Name	Reset	Width	Description
0x118	TRCSEQRSTEVR	—	32-bit	Sequencer Reset Control Register
0x11C	TRCSEQSTR	—	32-bit	Sequencer State Register
0x120	TRCEXTINSELR0	—	32-bit	External Input Select Register <n>
0x124	TRCEXTINSELR1	—	32-bit	External Input Select Register <n>
0x128	TRCEXTINSELR2	—	32-bit	External Input Select Register <n>
0x12C	TRCEXTINSELR3	—	32-bit	External Input Select Register <n>
0x140	TRCCNTRLDVR0	—	32-bit	Counter Reload Value Register <n>
0x144	TRCCNTRLDVR1	—	32-bit	Counter Reload Value Register <n>
0x150	TRCCNTCTLR0	—	32-bit	Counter Control Register <n>
0x154	TRCCNTCTLR1	—	32-bit	Counter Control Register <n>
0x160	TRCCNTVR0	—	32-bit	Counter Value Register <n>
0x164	TRCCNTVR1	—	32-bit	Counter Value Register <n>
0x180	TRCIDR8	—	32-bit	ID Register 8
0x184	TRCIDR9	—	32-bit	ID Register 9
0x188	TRCIDR10	—	32-bit	ID Register 10
0x18C	TRCIDR11	—	32-bit	ID Register 11
0x190	TRCIDR12	—	32-bit	ID Register 12
0x194	TRCIDR13	—	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	—	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	—	32-bit	ID Register 0
0x1E4	TRCIDR1	—	32-bit	ID Register 1
0x1E8	TRCIDR2	—	32-bit	ID Register 2
0x1EC	TRCIDR3	—	32-bit	ID Register 3
0x1F0	TRCIDR4	—	32-bit	ID Register 4
0x1F4	TRCIDR5	—	32-bit	ID Register 5
0x1F8	TRCIDR6	—	32-bit	ID Register 6
0x1FC	TRCIDR7	—	32-bit	ID Register 7
0x2A0 + (4 * n)	TRCSSCSR_n_	—	32-bit	Single-shot Comparator Control Status Register <n>
0x304	TRCOSLSR	—	32-bit	Trace OS Lock Status Register
0x310	TRCPDCR	—	32-bit	PowerDown Control Register
0x314	TRCPDSR	—	32-bit	PowerDown Status Register
0x680	TRCCIDCCTLR0	—	32-bit	Context Identifier Comparator Control Register 0
0x688	TRCVMIDCCTLR0	—	32-bit	Virtual Context Identifier Comparator Control Register 0
0xF00	TRCITCTRL	—	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	—	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	—	32-bit	Claim Tag Clear Register
0xFA8	TRCDEVAFF	—	64-bit	Device Affinity Register
0xFB0	TRCLAR	—	32-bit	Lock Access Register
0xFB4	TRCLSR	—	32-bit	Lock Status Register
0xFB8	TRCAUTHSTATUS	—	32-bit	Authentication Status Register

Offset	Name	Reset	Width	Description
0xFBC	TRCDEVARCH	–	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	–	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	–	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	–	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	–	32-bit	Device Type Register
0xFD0	TRCPIDR4	–	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	–	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	–	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	–	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	–	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	–	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	–	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	–	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	–	32-bit	Component Identification Register 0
0xFF4	TRCCIDR1	–	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	–	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	–	32-bit	Component Identification Register 3

B.7.1 TRCPRGCTLR, Programming Control Register

Enables the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x004

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-238: ext_trcprgctlr bit assignments

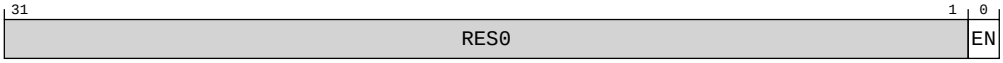


Table B-441: TRCPRGCTLR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	EN	Trace unit enable. **0b0** The trace unit is disabled. **0b1** The trace unit is enabled.	0b0

Accessibility

Must be programmed.

Component	Offset	Instance	Range
ETE	0x004	TRCPRGCTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.2 TRCSTATR, Trace Status Register

Returns the trace unit status.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component

ETE

Register offset

0x00C

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-239: ext_trcstatr bit assignments

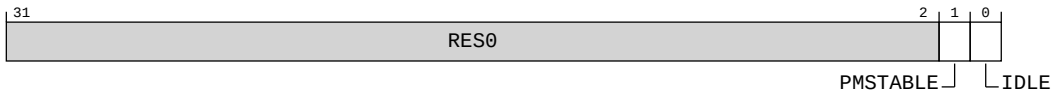


Table B-443: TRCSTATR bit descriptions

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**
[1]	PMSTABLE	Programmers' model stable. **0b0** The programmers' model is not stable. **0b1** The programmers' model is stable. **When Text("the trace unit is enabled")** Access to this field is: **UNKNOWN**/WI **Otherwise** Access to this field is: RO	x
[0]	IDLE	Idle status. **0b0** The trace unit is not idle. **0b1** The trace unit is idle.	x

Accessibility

Component	Offset	Instance	Range
ETE	0x00C	TRCSTATR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.3 TRCCONFIGR, Trace Configuration Register

Controls the tracing options.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x010

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-240: ext_trcconfigr bit assignments

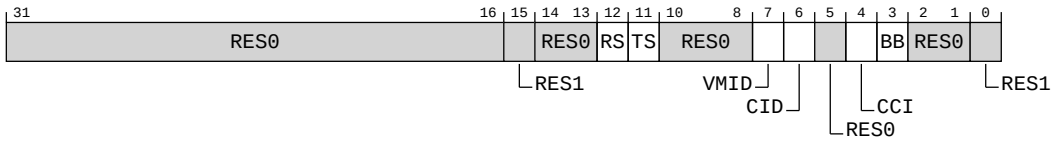


Table B-445: TRCCONFIGR bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15]	**RES1**	Reserved	**RES1**
[14:13]	**RES0**	Reserved	**RES0**
[12]	RS	Return stack control. **0b0** Return stack is disabled. **0b1** Return stack is enabled.	x
[11]	TS	Global timestamp tracing control. **0b0** Global timestamp tracing is disabled. **0b1** Global timestamp tracing is enabled.	x
[10:8]	**RES0**	Reserved	**RES0**
[7]	VMID	Virtual context identifier tracing control. **0b0** Virtual context identifier tracing is disabled. **0b1** Virtual context identifier tracing is enabled.	x
[6]	CID	Context identifier tracing control. **0b0** Context identifier tracing is disabled. **0b1** Context identifier tracing is enabled.	x
[5]	**RES0**	Reserved	**RES0**
[4]	CCI	Cycle counting instruction tracing control. **0b0** Cycle counting instruction tracing is disabled. **0b1** Cycle counting instruction tracing is enabled.	x

Bits	Name	Description	Reset
[3]	BB	Branch broadcasting control. **0b0** Branch broadcasting is disabled. **0b1** Branch broadcasting is enabled.	x
[2:1]	**RES0**	Reserved	**RES0**
[0]	**RES1**	Reserved	**RES1**

Access

Must always be programmed.

TRCCONFIGR.QE must be set to `0b00` if TRCCONFIGR.BB is not 0.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must always be programmed.

TRCCONFIGR.QE must be set to 0b00 if TRCCONFIGR.BB is not 0.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x010	TRCCONFIGR	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered() || !AllowExternalTraceAccess()
ERROR

Otherwise
RW

B.7.4 TRCAUXCTLR, Auxiliary 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

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-241: ext_trcauxctlr bit assignments

Table B-447: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

If this register is nonzero then it might cause the behavior of a trace unit to contradict this architecture specification. See the documentation of the specific implementation for information about the IMPLEMENTATION DEFINED support for this register.

Component	Offset	Instance	Range
ETE	0x018	TRCAUXCTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.5 TRCEVENTCTL0R, Event Control 0 Register

Controls the generation of ETEEvents.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x020

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-242: ext_trceventctl0r bit assignments

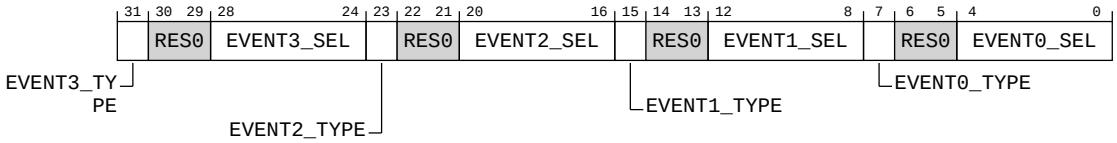


Table B-449: TRCEVENTCTL0R bit descriptions

Bits	Name	Description	Reset
[31]	EVENT3_TYPE	Chooses the type of Resource Selector. **0b0** A single Resource Selector. TRCEVENTCTL0R.EVENT3.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT3.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT3.SEL[4] is **RES0**.	x
[30:29]	**RES0**	Reserved	**RES0**
[28:24]	EVENT3_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT3.TYPE controls whether TRCEVENTCTL0R.EVENT3.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[3] == 1, then Event element 3 is generated in the instruction trace element stream.	5{x}
[23]	EVENT2_TYPE	Chooses the type of Resource Selector. **0b0** A single Resource Selector. TRCEVENTCTL0R.EVENT2.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT2.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT2.SEL[4] is **RES0**.	x
[22:21]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[20:16]	EVENT2_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT2.TYPE controls whether TRCEVENTCTL0R.EVENT2.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[2] == 1, then Event element 2 is generated in the instruction trace element stream.	5{x}
[15]	EVENT1_TYPE	Chooses the type of Resource Selector. **0b0** A single Resource Selector. TRCEVENTCTL0R.EVENT1.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT1.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT1.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	EVENT1_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT1.TYPE controls whether TRCEVENTCTL0R.EVENT1.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[1] == 1, then Event element 1 is generated in the instruction trace element stream.	5{x}
[7]	EVENT0_TYPE	Chooses the type of Resource Selector. **0b0** A single Resource Selector. TRCEVENTCTL0R.EVENT0.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCEVENTCTL0R.EVENT0.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCEVENTCTL0R.EVENT0.SEL[4] is **RES0**.	x

Bits	Name	Description	Reset
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	EVENT0_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCEVENTCTL0R.EVENT0.TYPE controls whether TRCEVENTCTL0R.EVENT0.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[0] == 1, then Event element 0 is generated in the instruction trace element stream.	5{x}

Access

Must be programmed if implemented.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if implemented.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x020	TRCEVENTCTL0R	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.6 TRCEVENTCTL1R, Event Control 1 Register

Controls the behavior of the ETEEvents that ext-TRCEVENTCTL0R selects.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x024

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-243: ext_trceventctl1r bit assignments

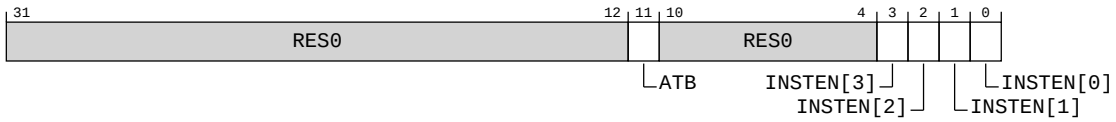


Table B-451: TRCEVENTCTL1R bit descriptions

Bits	Name	Description	Reset
[31:12]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[11]	ATB	AMBA Trace Bus (ATB) trigger enable. If a CoreSight ATB interface is implemented then when ETEEvent 0 occurs the trace unit sets: • ATID == 0x7D. • ATDATA to the value of ext-TRCTRACEIDR. If the width of ATDATA is greater than the width of ext-TRCTRACEIDR.TRACEID then the trace unit zeros the upper ATDATA bits. If ETEEvent 0 is programmed to occur based on program execution, such as an Address Comparator, the ATB trigger might not be inserted into the ATB stream at the same time as any trace generated by that program execution is output by the trace unit. Typically, the generated trace might be buffered in a trace unit which means that the ATB trigger would be output before the associated trace is output. If ETEEvent 0 is asserted multiple times in close succession, the trace unit is required to generate an ATB trigger for the first assertion, but might ignore one or more of the subsequent assertions. Arm recommends that the window in which ETEEvent 0 is ignored is limited only by the time taken to output an ATB trigger. **0b0** ATB trigger is disabled. **0b1** ATB trigger is enabled.	x
[10:4]	**RES0**	Reserved	**RES0**
[3:0]	INSTEN[<m>], bit[m], where m = 3 to 0	Event element control. **0b0** The trace unit does not generate an Event element <m>. **0b1** The trace unit generates an Event element <m>. This bit is **RES0** if m >= the number indicated by ext-TRCIDR0.NUMEVENT.	xxxx

Access

Must be programmed.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x024	TRCEVENTCTL1R	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.7 TRCRSR, Resources Status Register

Use this to set, or read, the status of the resources.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x028

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-244: ext_trcrsr bit assignments

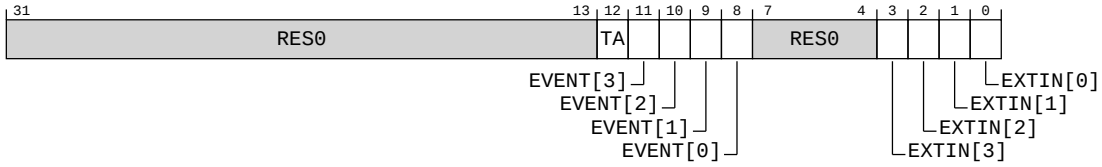


Table B-453: TRCRSR bit descriptions

Bits	Name	Description	Reset
[31:13]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[12]	TA	Tracing active. **0b0** Tracing is not active. **0b1** Tracing is active.	x
[11:8]	EVENT[<m>], bit[m], where m = 3 to 0	Untraced status of ETEEvents. **0b0** An ETEEvent <m> has not occurred. **0b1** An ETEEvent <m> has occurred while the resources were in the Paused state. This bit is **RES0** if ext-TRCIDR4.NUMRSPAIR == 0 || m > ext-TRCIDR0.NUMEVENT.	xxxx
[7:4]	**RES0**	Reserved	**RES0**
[3:0]	EXTIN[<m>], bit[m], where m = 3 to 0	The sticky status of the External Input Selectors. **0b0** An event selected by External Input Selector <m> has not occurred. **0b1** At least one event selected by External Input Selector <m> has occurred while the resources were in the Paused state. This bit is **RES0** if m >= ext-TRCIDR5.NUMEXTINSEL.	xxxx

Access

Must always be programmed.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

Accessibility

Must always be programmed.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Component	Offset	Instance	Range
ETE	0x028	TRCRSR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.8 TRCTSCTLR, Timestamp Control Register

Controls the insertion of global timestamps in the trace stream.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x030

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-245: ext_trctsctlr bit assignments

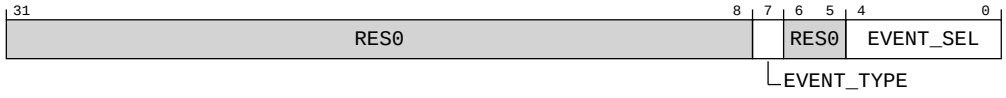


Table B-455: TRCTSCTLR bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7]	EVENT_TYPE	Chooses the type of Resource Selector. **0b0** A single Resource Selector. TRCTSCTLR.EVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCTSCTLR.EVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCTSCTLR.EVENT.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	EVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCTSCTLR.EVENT.TYPE controls whether TRCTSCTLR.EVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if ext-TRCCONFIGR.TS == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if ext-TRCCONFIGR.TS == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x030	TRCTSCTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.9 TRCSYNCPR, Synchronization Period Register

Controls how often trace protocol synchronization requests occur.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x034

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-246: ext_trcsyncpr bit assignments

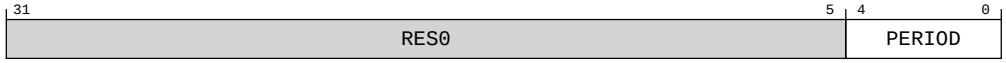


Table B-457: TRCSYNCPR bit descriptions

Bits	Name	Description	Reset
[31:5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4:0]	PERIOD	Defines the number of bytes of trace between each periodic trace protocol synchronization request. **0b00000** Trace protocol synchronization is disabled. **0b01000** Trace protocol synchronization request occurs after 2^8 bytes of trace. **0b01001** Trace protocol synchronization request occurs after 2^9 bytes of trace. **0b01010** Trace protocol synchronization request occurs after 2^{10} bytes of trace. **0b01011** Trace protocol synchronization request occurs after 2^{11} bytes of trace. **0b01100** Trace protocol synchronization request occurs after 2^{12} bytes of trace. **0b01101** Trace protocol synchronization request occurs after 2^{13} bytes of trace. **0b01110** Trace protocol synchronization request occurs after 2^{14} bytes of trace. **0b01111** Trace protocol synchronization request occurs after 2^{15} bytes of trace. **0b10000** Trace protocol synchronization request occurs after 2^{16} bytes of trace. **0b10001** Trace protocol synchronization request occurs after 2^{17} bytes of trace. **0b10010** Trace protocol synchronization request occurs after 2^{18} bytes of trace. **0b10011** Trace protocol synchronization request occurs after 2^{19} bytes of trace. **0b10100** Trace protocol synchronization request occurs after 2^{20} bytes of trace. Other values are reserved. If a reserved value is programmed into PERIOD, then the behavior of the synchronization period counter is CONSTRAINED UNPREDICTABLE and one of the following behaviors occurs: • No trace protocol synchronization requests are generated by this counter. • Trace protocol synchronization requests occur at the specified period. • Trace protocol synchronization requests occur at some other **UNKNOWN** period which can vary.	5{x}

Access

Must be programmed if ext-TRCIDR3.SYNCPR == 0.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if ext-TRCIDR3.SYNCPR == 0.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x034	TRCSYNCPR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.10 TRCCCCTLR, Cycle Count Control Register

Set the threshold value for cycle counting.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x038

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-247: ext_trcccctlr bit assignments

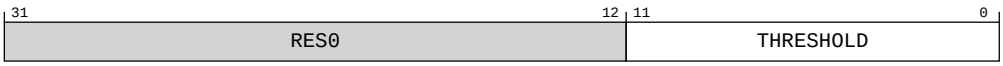


Table B-459: TRCCCCTLR bit descriptions

Bits	Name	Description	Reset
[31:12]	**RES0**	Reserved	**RES0**
[11:0]	THRESHOLD	Sets the threshold value for instruction trace cycle counting. The minimum threshold value that can be programmed into THRESHOLD is given in ext-TRCIDR3.CCITMIN. If the THRESHOLD value is smaller than the value in ext-TRCIDR3.CCITMIN then the behavior is **CONSTRAINED UNPREDICTABLE**. That is, cycle counts might or might not be included in the trace and the cycle count threshold is not known. Writing a value of zero when ext-TRCCONFIGR.CCI enables instruction trace cycle counting results in **CONSTRAINED UNPREDICTABLE** behavior. That is, cycle counts might or might not be included in the trace and the cycle count threshold is not known.	`12{x}`

Access

Must be programmed if ext-TRCCONFIGR.CCI == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if ext-TRCCONFIGR.CCI == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x038	TRCCCCTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.11 TRCBBCTLR, Branch Broadcast Control Register

Controls the regions in the memory map where branch broadcasting is active.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x03C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-248: ext_trcbbctlr bit assignments

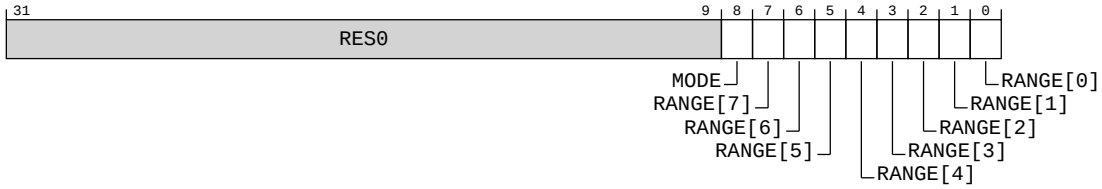


Table B-461: TRCBBCTLR bit descriptions

Bits	Name	Description	Reset
[31:9]	**RES0**	Reserved	**RES0**
[8]	MODE	Mode. **0b0** Exclude Mode. Branch broadcasting is not active for instructions in the address ranges defined by TRCBBCTLR.RANGE. If TRCBBCTLR.RANGE == 0x00 then branch broadcasting is active for all instructions. **0b1** Include Mode. Branch broadcasting is active for instructions in the address ranges defined by TRCBBCTLR.RANGE. If TRCBBCTLR.RANGE == 0x00 then the behavior of the trace unit is **CONSTRAINED UNPREDICTABLE**. That is, the trace unit might or might not consider any instructions to be in a branch broadcasting region.	x

Bits	Name	Description	Reset
[7:0]	RANGE[<m>], bit[m], where m = 7 to 0	Address range field. Selects whether Address Range Comparator <m> is used with branch broadcasting. **0b0** The address range that Address Range Comparator <m> defines, is not selected. **0b1** The address range that Address Range Comparator <m> defines, is selected. This bit is **RES0** if m >= ext-TRCIDR4.NUMACPAIRS.	8{x}

Access

Must be programmed if ext-TRCCONFIGR.BB == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if ext-TRCCONFIGR.BB == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x03C	TRCBBCTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.12 TRCTRACEIDR, Trace ID Register

Sets the trace ID for instruction trace.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0x040

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-249: ext_trctraceidr bit assignments

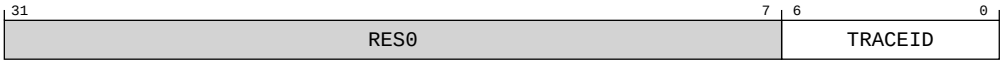


Table B-463: TRCTRACEIDR bit descriptions

Bits	Name	Description	Reset
[31:7]	**RES0**	Reserved	**RES0**
[6:0]	TRACEID	Trace ID field. Sets the trace ID value for instruction trace. The width of the field is indicated by the value of ext-TRCIDR5.TRACEIDSIZE. Unimplemented bits are **RES0**. If an implementation supports AMBA ATB, then: • The width of the field is 7 bits. • Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause **UNPREDICTABLE** behavior of the trace capture infrastructure. See the AMBA ATB Protocol Specification for information about which ATID values are reserved.	7{x}

Access

Must be programmed if implemented.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if implemented.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x040	TRCTRACEIDR	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered() || !AllowExternalTraceAccess()

ERROR

Otherwise

RW

B.7.13 TRCVICTLR, ViewInst Main Control Register

Controls instruction trace filtering.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x080

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-250: ext_trcvictlr bit assignments

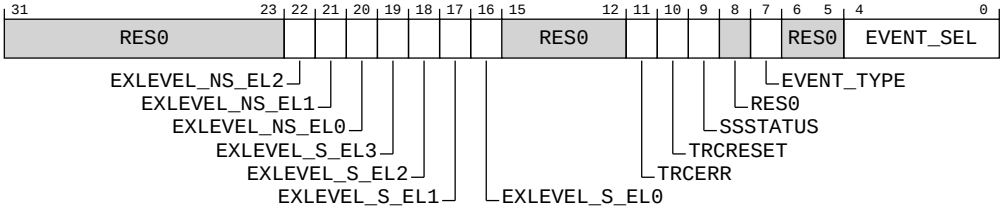


Table B-465: TRCVICTLR bit descriptions

Bits	Name	Description	Reset
[31:23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Filter instruction trace for EL2 in Non-secure state. **0b0** The trace unit generates instruction trace for EL2 in Non-secure state. **0b1** The trace unit does not generate instruction trace for EL2 in Non-secure state.	x
[21]	EXLEVEL_NS_EL1	Filter instruction trace for EL1 in Non-secure state. **0b0** The trace unit generates instruction trace for EL1 in Non-secure state. **0b1** The trace unit does not generate instruction trace for EL1 in Non-secure state.	x
[20]	EXLEVEL_NS_EL0	Filter instruction trace for EL0 in Non-secure state. **0b0** The trace unit generates instruction trace for EL0 in Non-secure state. **0b1** The trace unit does not generate instruction trace for EL0 in Non-secure state.	x
[19]	EXLEVEL_S_EL3	Filter instruction trace for EL3. **0b0** The trace unit generates instruction trace for EL3. **0b1** The trace unit does not generate instruction trace for EL3.	x
[18]	EXLEVEL_S_EL2	Filter instruction trace for EL2 in Secure state. **0b0** The trace unit generates instruction trace for EL2 in Secure state. **0b1** The trace unit does not generate instruction trace for EL2 in Secure state.	x
[17]	EXLEVEL_S_EL1	Filter instruction trace for EL1 in Secure state. **0b0** The trace unit generates instruction trace for EL1 in Secure state. **0b1** The trace unit does not generate instruction trace for EL1 in Secure state.	x
[16]	EXLEVEL_S_EL0	Filter instruction trace for EL0 in Secure state. **0b0** The trace unit generates instruction trace for EL0 in Secure state. **0b1** The trace unit does not generate instruction trace for EL0 in Secure state.	x
[15:12]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[11]	TRCERR	Controls the forced tracing of System Error exceptions. **0b0** Forced tracing of System Error exceptions is disabled. **0b1** Forced tracing of System Error exceptions is enabled.	x
[10]	TRCRESET	Controls the forced tracing of PE Resets. **0b0** Forced tracing of PE Resets is disabled. **0b1** Forced tracing of PE Resets is enabled.	x
[9]	SSSTATUS	ViewInst start/stop function status. **0b0** Stopped State. The ViewInst start/stop function is in the stopped state. **0b1** Started State. The ViewInst start/stop function is in the started state. Before software enables the trace unit, it must write to this field to set the initial state of the ViewInst start/stop function. If the ViewInst start/stop function is not used then set this field to 1. Arm recommends that the value of this field is set before each trace session begins. If the trace unit becomes disabled while a start point or stop point is still speculative, then the value of TRCVICTLR.SSSTATUS is **UNKNOWN** and might represent the result of a speculative start point or stop point. If software which is running on the PE being traced disables the trace unit, either by clearing ext-TRCPRGCTLR.EN or locking the OS Lock, Arm recommends that a DSB and an ISB instruction are executed before disabling the trace unit to prevent any start points or stop points being speculative at the point of disabling the trace unit. This procedure assumes that all start points or stop points occur before the barrier instructions are executed. The procedure does not guarantee that there are no speculative start points or stop points when disabling, although it helps minimize the probability. Access to this field is: RW	x
[8]	**RES0**	Reserved	**RES0**
[7]	EVENT_TYPE	Chooses the type of Resource Selector. **0b0** A single Resource Selector. TRCVICTLR.EVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCVICTLR.EVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCVICTLR.EVENT.SEL[4] is **RES0**.	x

Bits	Name	Description	Reset
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	EVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCVICTLR.EVENT.TYPE controls whether TRCVICTLR.EVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

Accessibility

Must be programmed.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Component	Offset	Instance	Range
ETE	0x080	TRCVICTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.14 TRCVIIECTLR, ViewInst Include/Exclude Control Register

Use this to select, or read, the Address Range Comparators for the ViewInst include/exclude function.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x084

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-251: ext_trcviiectlr bit assignments

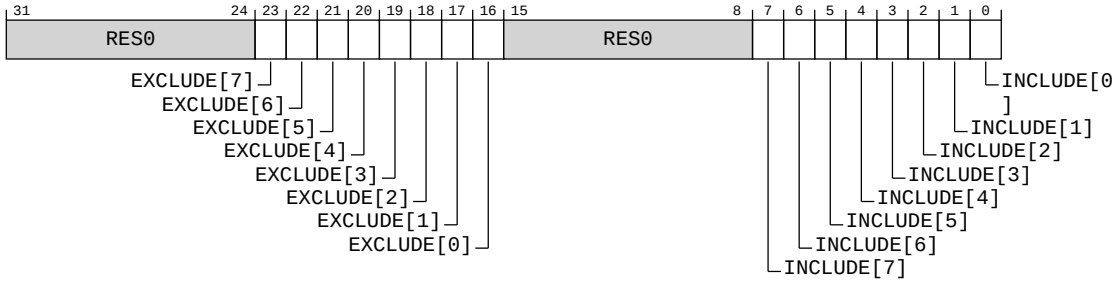


Table B-467: TRCVIIECTLR bit descriptions

Bits	Name	Description	Reset
[31:24]	**RES0**	Reserved	**RES0**
[23:16]	EXCLUDE[<m>], bit[m], where m = 7 to 0	Exclude Address Range Comparator <m>. Selects whether Address Range Comparator <m> is in use with the ViewInst exclude function. **0b0** The address range that Address Range Comparator <m> defines, is not selected for the ViewInst exclude function. **0b1** The address range that Address Range Comparator <m> defines, is selected for the ViewInst exclude function. This bit is **RES0** if m >= ext-TRCIDR4.NUMACPAIRS.	8{x}
[15:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:0]	INCLUDE[<m>], bit[m], where m = 7 to 0	Include Address Range Comparator <m>. Selects whether Address Range Comparator <m> is in use with the ViewInst include function. Selecting no comparators for the ViewInst include function indicates that all instructions are included by default. The ViewInst exclude function then indicates which ranges are excluded. **0b0** The address range that Address Range Comparator <m> defines, is not selected for the ViewInst include function. **0b1** The address range that Address Range Comparator <m> defines, is selected for the ViewInst include function. This bit is **RES0** if m >= ext-TRCIDR4.NUMACPAIRS.	8{x}

Access

Must be programmed if ext-TRCIDR4.NUMACPAIRS > 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if ext-TRCIDR4.NUMACPAIRS > 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x084	TRCVIIECTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.15 TRCVISSCTLR, ViewInst Start/Stop Control Register

Use this to select, or read, the Single Address Comparators for the ViewInst start/stop function.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x088

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-252: ext_trcvissctlr bit assignments

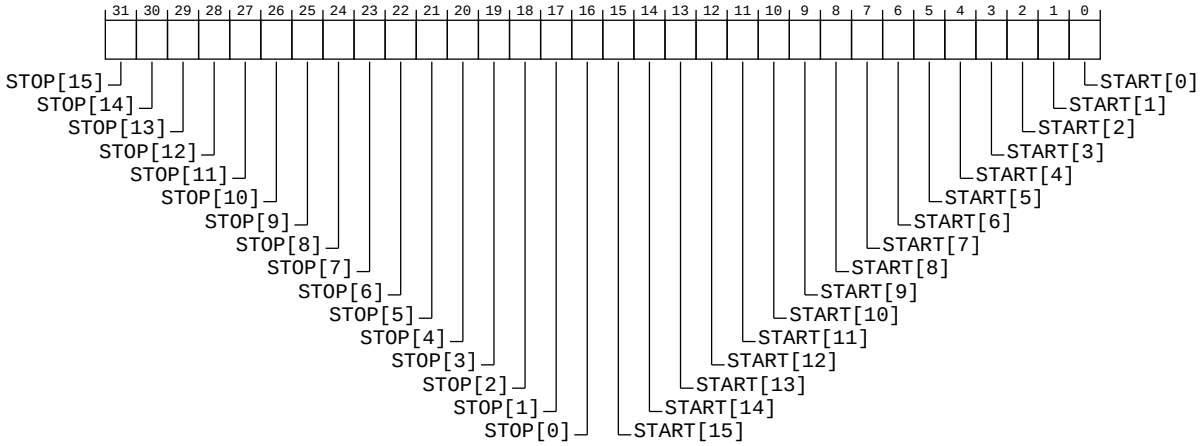


Table B-469: TRCVISSCTLR bit descriptions

Bits	Name	Description	Reset
[31:16]	STOP[<m>], bit[m], where m = 15 to 0	Selects whether Single Address Comparator <m> is used with the ViewInst start/stop function, for the purpose of stopping trace. **0b0** The Single Address Comparator <m>, is not selected as a stop resource. **0b1** The Single Address Comparator <m>, is selected as a stop resource. This bit is **RES0** if m >= 2 × ext-TRCIDR4.NUMACPAIRS.	16{x}
[15:0]	START[<m>], bit[m], where m = 15 to 0	Selects whether Single Address Comparator <m> is used with the ViewInst start/stop function, for the purpose of starting trace. **0b0** The Single Address Comparator <m>, is not selected as a start resource. **0b1** The Single Address Comparator <m>, is selected as a start resource. This bit is **RES0** if m >= 2 × ext-TRCIDR4.NUMACPAIRS.	16{x}

Access

Must be programmed if ext-TRCIDR4.NUMACPAIRS > 0b0000.

For any 2 comparators selected for the ViewInst start/stop function, the comparator containing the lower address must be a lower numbered comparator.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if ext-TRCIDR4.NUMACPAIRS > 0b0000.

For any 2 comparators selected for the ViewInst start/stop function, the comparator containing the lower address must be a lower numbered comparator.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x088	TRCVISSCTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.16 TRCSEQEVR0, Sequencer State Transition Control Register <n>

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x100

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-253: ext_trcseqevr0 bit assignments

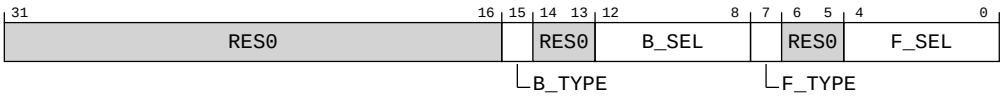


Table B-471: TRCSEQEVR0 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15]	B_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == `0x14` then when event `0x14` occurs, the Sequencer moves from state 3 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	B_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Backward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}
[7]	F_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == `0x12` then when event `0x12` occurs, the Sequencer moves from state 1 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4:0]	F_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Forward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	`5{x}`

Access

Must be programmed if TRCRSCTLR<a>.GROUP == `0b0010` and TRCRSCTLR<a>.SEQUENCER != `0b0000`.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x100	TRCSEQEVR0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.17 TRCSEQEVR1, Sequencer State Transition Control Register <n>

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x104

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-254: ext_trcseqevr1 bit assignments

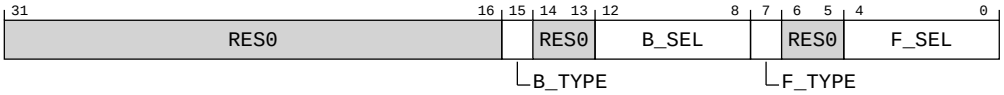


Table B-473: TRCSEQEVR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15]	B_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == `0x14` then when event `0x14` occurs, the Sequencer moves from state 3 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is **RES0**.	x

Bits	Name	Description	Reset
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	B_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Backward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}
[7]	F_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == 0x12 then when event 0x12 occurs, the Sequencer moves from state 1 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	F_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Forward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x104	TRCSEQEVR1	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.18 TRCSEQEVR2, Sequencer State Transition Control Register <n>

Moves the Sequencer state:

- Backwards, from state n+1 to state n when a programmed resource event occurs.
- Forwards, from state n to state n+1 when a programmed resource event occurs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x108

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-255: ext_trcseqevr2 bit assignments

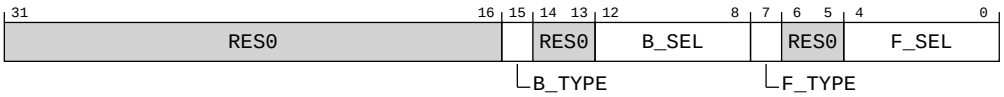


Table B-475: TRCSEQEVR2 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15]	B_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the backward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n+1 to state n. For example, if TRCSEQEVR2.B.SEL == `0x14` then when event `0x14` occurs, the Sequencer moves from state 3 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.B.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.B.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.B.SEL[4] is **RES0**.	`x`
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	B_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.B.TYPE controls whether TRCSEQEVR<n>.B.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Backward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	`5{x}`

Bits	Name	Description	Reset
[7]	F_TYPE	Chooses the type of Resource Selector. Backward field. Defines whether the forward resource event is a single Resource Selector or a Resource Selector pair. When the resource event occurs then the Sequencer state moves from state n to state n+1. For example, if TRCSEQEVR1.F.SEL == 0x12 then when event 0x12 occurs, the Sequencer moves from state 1 to state 2. **0b0** A single Resource Selector. TRCSEQEVR<n>.F.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQEVR<n>.F.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQEVR<n>.F.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	F_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQEVR<n>.F.TYPE controls whether TRCSEQEVR<n>.F.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Forward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x108	TRCSEQEVR2	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.19 TRCSEQRSTEVR, Sequencer Reset Control Register

Moves the Sequencer to state 0 when a programmed resource event occurs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x118

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-256: ext_trcseqrstevr bit assignments

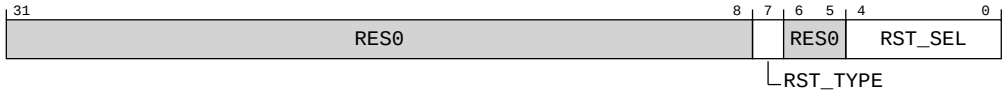


Table B-477: TRCSEQRSTEVR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7]	RST_TYPE	Chooses the type of Resource Selector. **0b0** A single Resource Selector. TRCSEQRSTEVR.RST.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCSEQRSTEVR.RST.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCSEQRSTEVR.RST.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**
[4:0]	RST_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCSEQRSTEVR.RST.TYPE controls whether TRCSEQRSTEVR.RST.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x118	TRCSEQRSTEVR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.20 TRCSEQSTR, Sequencer State Register

Use this to set, or read, the Sequencer state.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x11C

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-257: ext_trcseqstr bit assignments

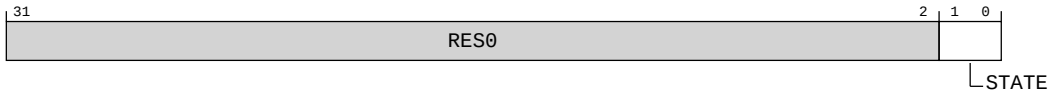


Table B-479: TRCSEQSTR bit descriptions

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1:0]	STATE	Set or returns the state of the Sequencer. **0b00** State 0. **0b01** State 1. **0b10** State 2. **0b11** State 3.	xx

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Component	Offset	Instance	Range
ETE	0x11C	TRCSEQSTR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.21 TRCEXTINSELR0, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x120

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-258: ext_trcextinselr0 bit assignments

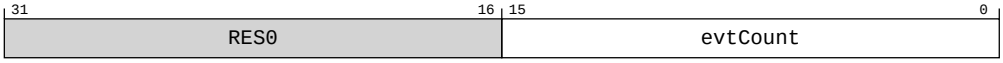


Table B-481: TRCEXTINSELR0 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == `0b0000` and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x120	TRCEXTINSELR0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.22 TRCEXTINSELR1, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x124

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-259: ext_trcextinselr1 bit assignments

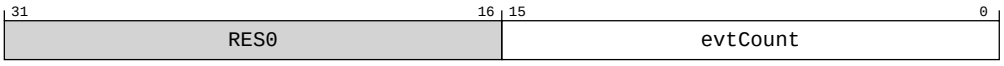


Table B-483: TRCEXTINSELR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == `0b0000` and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x124	TRCEXTINSELR1	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.23 TRCEXTINSELR2, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x128

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-260: ext_trcextinselr2 bit assignments

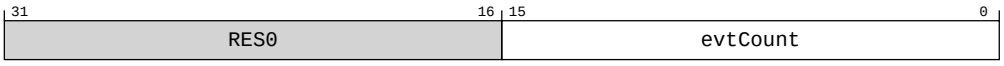


Table B-485: TRCEXTINSELR2 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == `0b0000` and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x128	TRCEXTINSELR2	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.24 TRCEXTINSELR3, External Input Select Register <n>

Use this to set, or read, which External Inputs are resources to the trace unit.

The name TRCEXTINSELR is an alias of TRCEXTINSELR0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x12C

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-261: ext_trcextinselr3 bit assignments

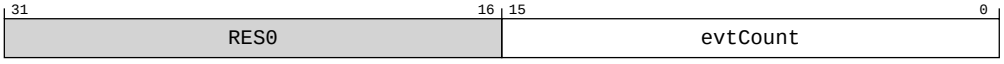


Table B-487: TRCEXTINSELR3 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:0]	evtCount	PMU event to select. The event number as defined by the Arm ARM. Software must program this field with a PMU event that is supported by the PE being programmed. There are three ranges of PMU event numbers: • PMU event numbers in the range `0x0000` to `0x003F` are common architectural and microarchitectural events. • PMU event numbers in the range `0x0040` to `0x00BF` are Arm recommended common architectural and microarchitectural PMU events. • PMU event numbers in the range `0x00C0` to `0x03FF` are IMPLEMENTATION DEFINED PMU events. If evtCount is programmed to a PMU event that is reserved or not supported by the PE, the behavior depends on the PMU event type: • For the range `0x0000` to `0x003F`, then the PMU event is not active, and the value returned by a direct or external read of the evtCount field is the value written to the field. • For IMPLEMENTATION DEFINED PMU events, it is UNPREDICTABLE what PMU event, if any, is counted, and the value returned by a direct or external read of the evtCount field is **UNKNOWN**. UNPREDICTABLE means the PMU event must not expose privileged information. Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include a PMU event from a set of common **IMPLEMENTATION DEFINED** PMU events, then no PMU event is counted and the value read back on evtCount is the value written.	`16{x}`

Access

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == `0b0000` and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if any of the following is true: TRCRSCTLR<a>.GROUP == 0b0000 and TRCRSCTLR<a>.EXTIN[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x12C	TRCEXTINSELR3	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.25 TRCCNTRLDVR0, Counter Reload Value Register <n>

This sets or returns the reload count value for Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x140

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-262: ext_trccntrldvr0 bit assignments

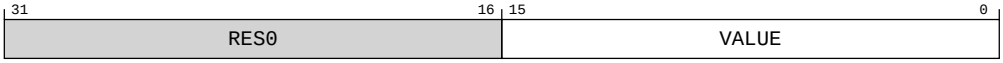


Table B-489: TRCCNTRLDVR0 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	VALUE	Contains the reload value for Counter <n>. When a reload event occurs for Counter <n> then the trace unit copies the VALUE<n> field into Counter <n>.	`16{x}`

Access

Must be programmed if TRCRSCTLR<a>.GROUP == `0b0010` and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x140	TRCCNTRLDVR0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.26 TRCCNTRLDVR1, Counter Reload Value Register <n>

This sets or returns the reload count value for Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x144

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-263: ext_trccntrldvr1 bit assignments

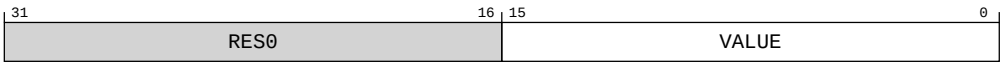


Table B-491: TRCCNTRLDVR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	VALUE	Contains the reload value for Counter <n>. When a reload event occurs for Counter <n> then the trace unit copies the VALUE<n> field into Counter <n>.	`16{x}`

Access

Must be programmed if TRCRSCTLR<a>.GROUP == `0b0010` and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x144	TRCCNTRLDVR1	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.27 TRCCNTCTLR0, Counter Control Register <n>

Controls the operation of Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x150

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-264: ext_trccntctlr0 bit assignments

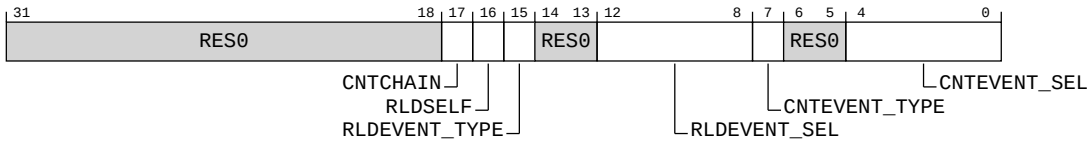


Table B-493: TRCCNTCTLR0 bit descriptions

Bits	Name	Description	Reset
[31:18]	**RES0**	Reserved	**RES0**
[17]	CNTCHAIN	For TRCCNTCTLR3 and TRCCNTCTLR1, this field controls whether the Counter decrements when a reload event occurs for Counter <n-1>. **0b0** The Counter does not decrement when a reload event for Counter <n-1> occurs. **0b1** Counter <n> decrements when a reload event for Counter <n-1> occurs. This concatenates Counter <n> and Counter <n-1>, to provide a larger count value. CNTCHAIN is not implemented for TRCCNTCTLR0 and TRCCNTCTLR2.	x
[16]	RLDSELF	Controls whether a reload event occurs for the Counter, when the Counter reaches zero. **0b0** Normal mode. The Counter is in Normal mode. **0b1** Self-reload mode. The Counter is in Self-reload mode.	x

Bits	Name	Description	Reset
[15]	RLDEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes a reload event for Counter <n>. **0b0** A single Resource Selector. TRCCNTCTLR<n>.RLDEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.RLDEVENT.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	RLDEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.TYPE controls whether TRCCNTCTLR<n>.RLDEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes a reload event for Counter <n>. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}
[7]	CNTEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes Counter <n> to decrement. **0b0** A single Resource Selector. TRCCNTCTLR<n>.CNTEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.CNTEVENT.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4:0]	CNTEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.TYPE controls whether TRCCNTCTLR<n>.CNTEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes Counter <n> to decrement. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x150	TRCCNTCTLR0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.28 TRCCNTCTLR1, Counter Control Register <n>

Controls the operation of Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x154

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-265: ext_trccntctlr1 bit assignments

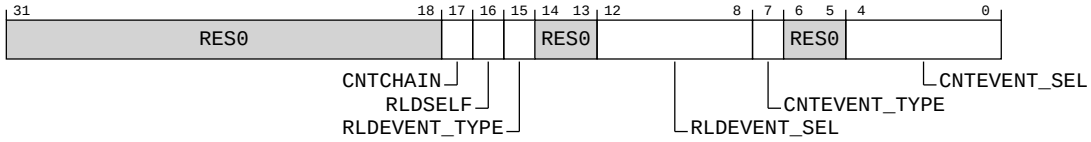


Table B-495: TRCCNTCTLR1 bit descriptions

Bits	Name	Description	Reset
[31:18]	**RES0**	Reserved	**RES0**
[17]	CNTCHAIN	For TRCCNTCTLR3 and TRCCNTCTLR1, this field controls whether the Counter decrements when a reload event occurs for Counter <n-1>. **0b0** The Counter does not decrement when a reload event for Counter <n-1> occurs. **0b1** Counter <n> decrements when a reload event for Counter <n-1> occurs. This concatenates Counter <n> and Counter <n-1>, to provide a larger count value. CNTCHAIN is not implemented for TRCCNTCTLR0 and TRCCNTCTLR2.	x
[16]	RLDSELF	Controls whether a reload event occurs for the Counter, when the Counter reaches zero. **0b0** Normal mode. The Counter is in Normal mode. **0b1** Self-reload mode. The Counter is in Self-reload mode.	x

Bits	Name	Description	Reset
[15]	RLDEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes a reload event for Counter <n>. **0b0** A single Resource Selector. TRCCNTCTLR<n>.RLDEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.RLDEVENT.SEL[4] is **RES0**.	x
[14:13]	**RES0**	Reserved	**RES0**
[12:8]	RLDEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.RLDEVENT.TYPE controls whether TRCCNTCTLR<n>.RLDEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes a reload event for Counter <n>. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}
[7]	CNTEVENT_TYPE	Chooses the type of Resource Selector. Selects an event, that when it occurs causes Counter <n> to decrement. **0b0** A single Resource Selector. TRCCNTCTLR<n>.CNTEVENT.SEL[4:0] selects the single Resource Selector, from 0-31, used to activate the resource event. **0b1** A Boolean-combined pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.SEL[3:0] selects the Resource Selector pair, from 0-15, that has a Boolean function that is applied to it whose output is used to activate the resource event. TRCCNTCTLR<n>.CNTEVENT.SEL[4] is **RES0**.	x
[6:5]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[4:0]	CNTEVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. TRCCNTCTLR<n>.CNTEVENT.TYPE controls whether TRCCNTCTLR<n>.CNTEVENT.SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Selects an event, that when it occurs causes Counter <n> to decrement. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state. Selecting Resource Selector pair 0 using this field is **UNPREDICTABLE**, and the resource event might fire or might not fire when the resources are not in the Paused state.	5{x}

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x154	TRCCNTCTLR1	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.29 TRCCNTVR0, Counter Value Register <n>

This sets or returns the value of Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x160

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-266: ext_trccntvr0 bit assignments

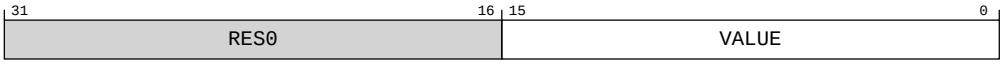


Table B-497: TRCCNTVR0 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	VALUE	Contains the count value of Counter.	`16{x}`

Access

Must be programmed if TRCRSCTLR<a>.GROUP == `0b0010` and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Component	Offset	Instance	Range
ETE	0x160	TRCCNTVR0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.30 TRCCNTVR1, Counter Value Register <n>

This sets or returns the value of Counter <n>.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x164

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-267: ext_trccntvr1 bit assignments

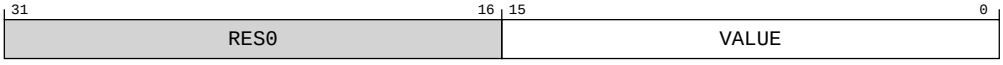


Table B-499: TRCCNTVR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	VALUE	Contains the count value of Counter.	`16{x}`

Access

Must be programmed if TRCRSCTLR<a>.GROUP == `0b0010` and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Component	Offset	Instance	Range
ETE	0x164	TRCCNTVR1	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.31 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-268: ext_trcidr8 bit assignments

Table B-501: 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.	32{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x180	TRCIDR8	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.32 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-269: ext_trcidr9 bit assignments

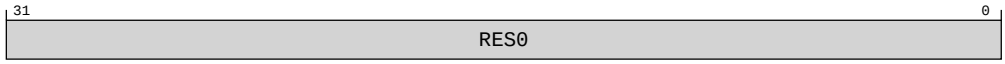


Table B-503: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x184	TRCIDR9	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.33 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-270: ext_trcidr10 bit assignments

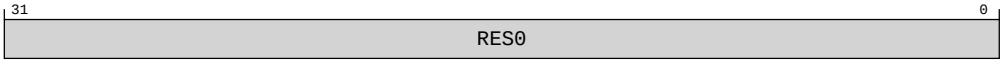


Table B-505: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x188	TRCIDR10	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.34 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

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-271: ext_trcidr11 bit assignments

Table B-507: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
ETE	0x18C	TRCIDR11	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.35 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-272: ext_trcidr12 bit assignments

Table B-509: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
ETE	0x190	TRCIDR12	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.36 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

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-273: ext_trcidr13 bit assignments

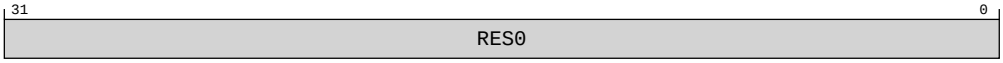


Table B-511: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x194	TRCIDR13	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.37 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

Access type

RO

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-274: ext_trcimspec0 bit assignments

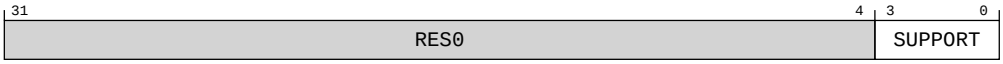


Table B-513: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No **IMPLEMENTATION DEFINED** features are supported.	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1C0	TRCIMSPEC0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.38 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

Access type
See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-275: ext_trcidr0 bit assignments

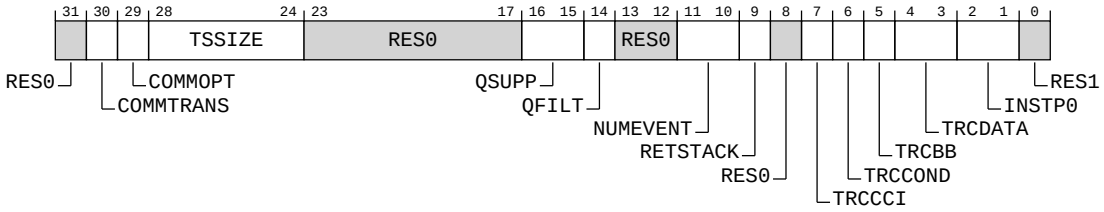


Table B-515: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	x
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1. **When UInt(ext-TRCIDR8.MAXSPEC) == 0x0** Access to this field is: **RAO/WI** **Otherwise** Access to this field is: RO	x
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	5{x}
[23:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	xx
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	x
[13:12]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	xx
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **0b1** Return stack implemented.	x
[8]	**RES0**	Reserved	**RES0**
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **0b1** Cycle counting implemented.	x
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b0** Conditional instruction tracing not implemented.	x
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **0b1** Branch broadcasting implemented.	x
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Data tracing not implemented.	xx
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. **0b00** Load and store instructions are not P0 instructions.	xx
[0]	**RES1**	Reserved	**RES1**

Accessibility

Component	Offset	Instance	Range
ETE	0x1E0	TRCIDR0	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.39 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-276: ext_trcidr1 bit assignments

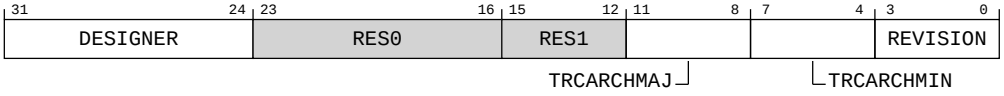


Table B-517: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. **0b01000001** Arm Limited	8{x}
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	**RES1**	Reserved	**RES1**

Bits	Name	Description	Reset
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	xxxx
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	xxxx
[3:0]	REVISION	Implementation revision that identifies the revision of the trace and OS Lock registers. **0b0001** Revision 1	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1E4	TRCIDR1	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.40 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

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-277: ext_trcidr2 bit assignments

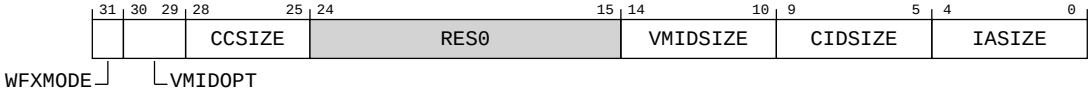


Table B-519: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions. **0b1** `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions.	x
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is **RES1**.	xx
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0b0000** The cycle counter is 12 bits in length.	xxxx
[24:15]	**RES0**	Reserved	**RES0**
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **0b00100** 32-bit Virtual context identifier size.	5{x}
[9:5]	CIDSIZE	Indicates the Context identifier size. **0b00100** 32-bit Context identifier size.	5{x}
[4:0]	IASIZE	Virtual instruction address size. **0b01000** Maximum of 64-bit instruction address size.	5{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x1E8	TRCIDR2	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.41 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-278: ext_trcidr3 bit assignments

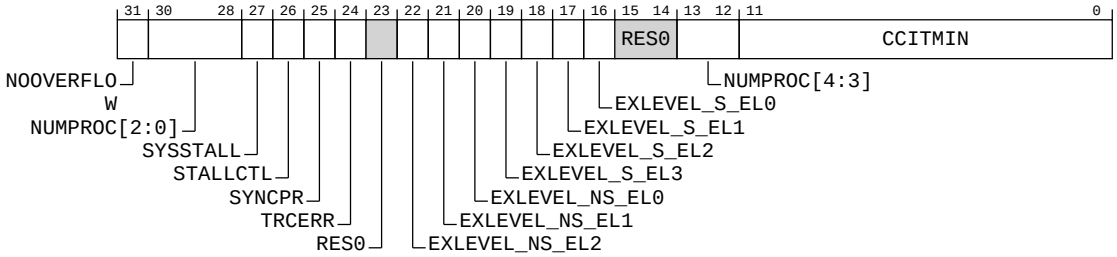


Table B-521: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	x
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	x
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	x
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** ext-TRCSYNCPR is read/write so software can change the synchronization period.	x
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **0b1** Forced tracing of System Error exceptions is implemented.	x
[23]	**RES0**	Reserved	**RES0**
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. **0b1** Non-secure EL2 is implemented.	x
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. **0b1** Non-secure EL1 is implemented.	x
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 implemented. **0b1** Non-secure EL0 is implemented.	x
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. **0b1** EL3 is implemented.	x
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. **0b1** Secure EL2 is implemented.	x
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. **0b1** Secure EL1 is implemented.	x
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 implemented. **0b1** Secure EL0 is implemented.	x
[15:14]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	5{x}
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in ext-TRCCCCTLR.THRESHOLD. Reserved at 0x004.	12{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x1EC	TRCIDR3	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.42 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

Access type
See bit descriptions

Reset value
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-279: ext_trcidr4 bit assignments

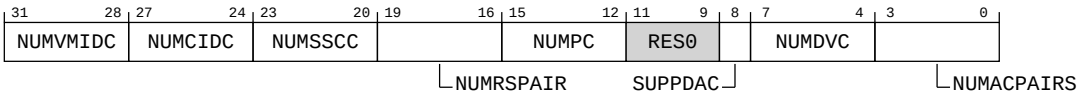


Table B-523: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Virtual Context Identifier Comparator.	xxxx
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	xxxx
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The implementation has one Single-shot Comparator Control.	xxxx
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	xxxx
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	xxxx
[11:9]	**RES0**	Reserved	**RES0**
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** Data address comparisons not implemented.	x
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** No data value comparators implemented.	xxxx
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The implementation has four Address Comparator pairs.	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1F0	TRCIDR4	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.43 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-280: ext_trcidr5 bit assignments

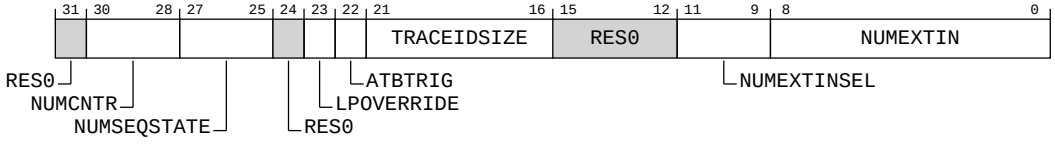


Table B-525: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **0b010** Two Counters implemented.	xxx
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **0b100** Four Sequencer states are implemented.	xxx
[24]	**RES0**	Reserved	**RES0**
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **0b0** The trace unit does not support Low-power Override Mode.	x
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **0b1** The implementation supports ATB triggers.	x
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **0b000111** The implementation supports a 7-bit trace ID.	6{x}
[15:12]	**RES0**	Reserved	**RES0**
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. **0b100** 4 External Input Selector resources are available.	xxx
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **0b111111111** Unified PMU event selection. All other values are reserved.	9{x}

Accessibility

Component	Offset	Instance	Range
ETE	0x1F4	TRCIDR5	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.44 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

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-281: ext_trcidr6 bit assignments

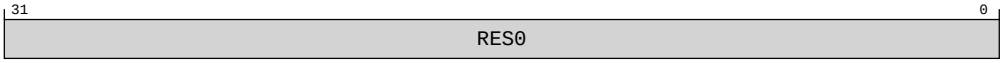


Table B-527: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x1F8	TRCIDR6	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise
RO

B.7.45 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

Access type
See bit descriptions

Reset value
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note
Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-282: ext_trcidr7 bit assignments

Table B-529: TRCIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
ETE	0x1FC	TRCIDR7	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.46 TRCSSCSR<n>, Single-shot Comparator Control Status Register <n> , n = 0 - 7

Returns the status of the corresponding Single-shot Comparator Control.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x2A0 + (4 * n)

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-283: ext_trcsscsr_n_ bit assignments

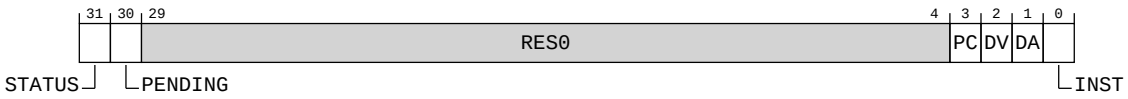


Table B-531: TRCSSCSR<n> bit descriptions

Bits	Name	Description	Reset
[31]	STATUS	Single-shot Comparator Control status. Indicates if any of the comparators selected by this Single-shot Comparator control have matched. The selected comparators are defined by ext-TRCSSCCR<n>.ARC, ext-TRCSSCCR<n>.SAC, and ext-TRCSSPCICR<n>.PC. **0b0** No match has occurred. When the first match occurs, this field takes a value of 1. It remains at 1 until explicitly modified by a write to this register. **0b1** One or more matches has occurred. If ext-TRCSSCCR<n>.RST == 0 then: • There is only one match and no more matches are possible. • Software must reset this field to 0 to re-enable the Single-shot Comparator Control.	x
[30]	PENDING	Single-shot pending status. The Single-shot Comparator Control fired while the resources were in the Paused state. **0b0** No match has occurred. **0b1** One or more matches has occurred.	x
[29:4]	**RES0**	Reserved	**RES0**
[3]	PC	PE Comparator Input support. Indicates if the Single-shot Comparator Control supports PE Comparator Inputs. **0b0** This Single-shot Comparator Control does not support PE Comparator Inputs. Selecting any PE Comparator Inputs using the associated ext-TRCSSPCICR<n> results in **CONSTRAINED UNPREDICTABLE** behavior of the Single-shot Comparator Control resource. The Single-shot Comparator Control might match unexpectedly or might not match. **0b1** This Single-shot Comparator Control supports PE Comparator Inputs.	x
[2]	DV	Data value comparator support. Data value comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** This Single-shot Comparator Control does not support data value comparisons. **0b1** This Single-shot Comparator Control supports data value comparisons. This field reads as 0.	x
[1]	DA	Data Address Comparator support. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** This Single-shot Comparator Control does not support data address comparisons. **0b1** This Single-shot Comparator Control supports data address comparisons. This field reads as 0.	x

Bits	Name	Description	Reset
[0]	INST	Instruction Address Comparator support. Indicates if the Single-shot Comparator Control supports instruction address comparisons. **0b0** This Single-shot Comparator Control does not support instruction address comparisons. **0b1** This Single-shot Comparator Control supports instruction address comparisons. This field reads as 1.	x

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 1.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Reads from this register might return an **UNKNOWN** value if the trace unit is not in either of the Idle or Stable states.

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 1.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Reads from this register might return an UNKNOWN value if the trace unit is not in either of the Idle or Stable states.

Component	Offset	Instance	Range
ETE	0x2A0 + (4 * n)	TRCSSCSR<n>	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.47 TRCOSLSR, Trace OS Lock Status Register

Returns the status of the Trace OS Lock.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x304

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-284: ext_trcoslsr bit assignments

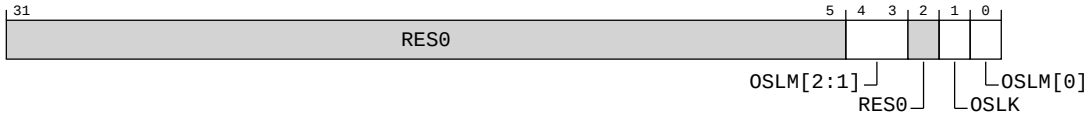


Table B-533: TRCOSLSR bit descriptions

Bits	Name	Description	Reset
[31:5]	**RES0**	Reserved	**RES0**
[4:3, 0]	OSLM	OS Lock model. **0b000** Trace OS Lock is not implemented. **0b010** Trace OS Lock is implemented. **0b100** Trace OS Lock is not implemented, and the trace unit is controlled by the PE OS Lock. All other values are reserved. This field reads as 0b100.	xxx
[2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1]	OSLK	OS Lock status. **0b0** The OS Lock is unlocked. **0b1** The OS Lock is locked. Note that this field indicates the state of the PE OS Lock.	x

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0x304	TRCOSLSR	None

This interface is accessible as follows:

When !AllowExternalTraceAccess() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.48 TRCPDCR, PowerDown Control Register

Requests the system to provide power to the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0x310

Access type
See bit descriptions

Reset value
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-285: ext_trcpdcr bit assignments

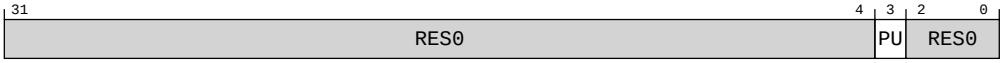


Table B-535: TRCPDCR bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3]	PU	Power Up Request. **0b0** The system can remove power from the trace unit core power domain, or requests for power to the trace unit core power domain are implemented outside of the trace unit. **0b1** The system must provide power to the trace unit core power domain. This field is **RES0**.	x
[2:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0x310	TRCPDCR	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.49 TRCPDSR, PowerDown Status Register

Indicates the power status of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x314

Access type

RAOWI

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx11

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-286: ext_trcpdsr bit assignments

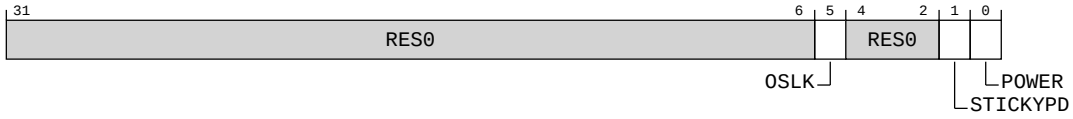


Table B-537: TRCPDSR bit descriptions

Bits	Name	Description	Reset
[31:6]	**RES0**	Reserved	**RES0**
[5]	OSLK	OS Lock Status. **0b0** The OS Lock is unlocked. **0b1** The OS Lock is locked. Note that this field indicates the state of the PE OS Lock.	x
[4:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1]	STICKYPD	Sticky powerdown status. Indicates whether the trace register state is valid. **0b0** The state of ext-TRCOSLSR and the trace registers are valid. **0b1** The state of ext-TRCOSLSR and the trace registers might not be valid. This field is set to 1 if the power to the trace unit core power domain is removed and the trace unit register state is not valid. The STICKYPD field is read-sensitive. On a read of the TRCPDSR, this field is cleared to 0 after the register has been read.	0b1
[0]	POWER	Power Status. **0b0** The trace unit core power domain is not powered. All trace unit registers are not accessible and they all return an error response. **0b1** The trace unit core power domain is powered. Trace unit registers are accessible.	0b1

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0x314	TRCPDSR	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.50 TRCCIDCCTLR0, Context Identifier Comparator Control Register 0

Contains Context identifier mask values for the ext-TRCCIDCVR<n> registers, for n = 0 to 3.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x680

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-287: ext_trccidcctlr0 bit assignments

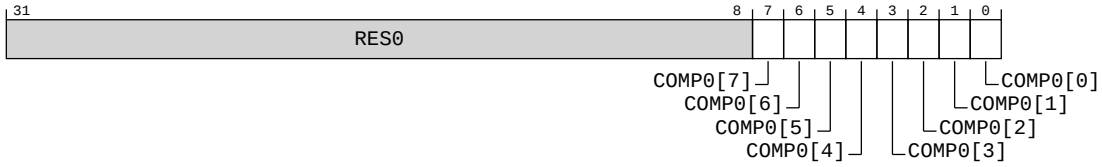


Table B-539: TRCCIDCCTLR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	COMP0[<m>], bit[m], where m = 7 to 0	TRCCIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0. **0b0** The trace unit includes TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison. **0b1** The trace unit ignores TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison. This bit is **RES0** if m >= ext-TRCIDR2.CIDSIZE.	8{x}

Access

If software uses the ext-TRCCIDCVR<n> registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in ext-TRCCIDCVR<n> to 0x00.

If any bit is 1 and the relevant byte in ext-TRCCIDCVR<n> is not `0x00`, the behavior of the Context Identifier Comparator is **CONSTRAINED UNPREDICTABLE**. In this scenario the comparator might match unexpectedly or might not match.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

If software uses the ext-TRCCIDCVR<n> registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in ext-TRCCIDCVR<n> to 0x00.

If any bit is 1 and the relevant byte in ext-TRCCIDCVR<n> is not 0x00, the behavior of the Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x680	TRCCIDCCTLR0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.51 TRCVMIDCCTLR0, Virtual Context Identifier Comparator Control Register 0

Virtual Context Identifier Comparator mask values for the ext-TRCVMIDCVR<n> registers, where n=0-3.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x688

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-288: ext_trcvmidcctlr0 bit assignments

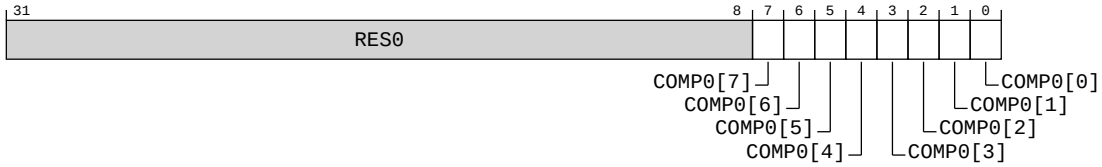


Table B-541: TRCVMIDCCTLR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	COMP0[<m>], bit[m], where m = 7 to 0	TRCVMIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0. **0b0** The trace unit includes TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison. **0b1** The trace unit ignores TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison. This bit is **RES0** if m >= ext-TRCIDR2.VMIDSIZE.	8{x}

Access

If software uses the ext-TRCVMIDCVR<n> registers, where n=0-3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in ext-TRCVMIDCVR<n> to `0x00`.

If any bit is 1 and the relevant byte in ext-TRCVMIDCVR<n> is not `0x00`, the behavior of the Virtual Context Identifier Comparator is **CONSTRAINED UNPREDICTABLE**. In this scenario the comparator might match unexpectedly or might not match.

Writes are **CONSTRAINED UNPREDICTABLE** if the trace unit is not in the Idle state.

Accessibility

If software uses the ext-TRCVMIDCVR<n> registers, where n=0-3, then it must program this register.

If software sets a mask bit to 1 then it must program the relevant byte in ext-TRCVMIDCVR<n> to 0x00.

If any bit is 1 and the relevant byte in ext-TRCVMIDCVR<n> is not 0x00, the behavior of the Virtual Context Identifier Comparator is CONSTRAINED UNPREDICTABLE. In this scenario the comparator might match unexpectedly or might not match.

Writes are CONSTRAINED UNPREDICTABLE if the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0x688	TRCVMIDCCTLR0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.52 TRCITCTRL, Integration Mode Control Register

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xF00

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-289: ext_trcitctrl bit assignments

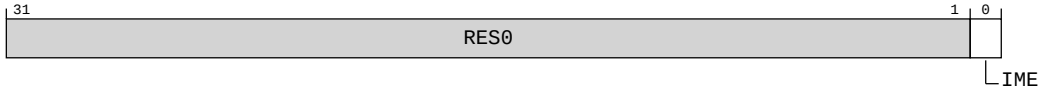


Table B-543: TRCITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	IME	Integration Mode Enable. **0b0** Component functional mode. **0b1** Component integration mode. Support for topology detection and integration testing is enabled.	x

Accessibility

External debugger accesses to this register are IMPLEMENTATION DEFINED when the trace unit is not in the Idle state.

Component	Offset	Instance	Range
ETE	0xF00	TRCITCTRL	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.53 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

Access type

RAOW1S

Reset value

1111 1111 1111 1111 1111 1111 1111 1111

Bit descriptions

Figure B-290: ext_trcclaimset bit assignments

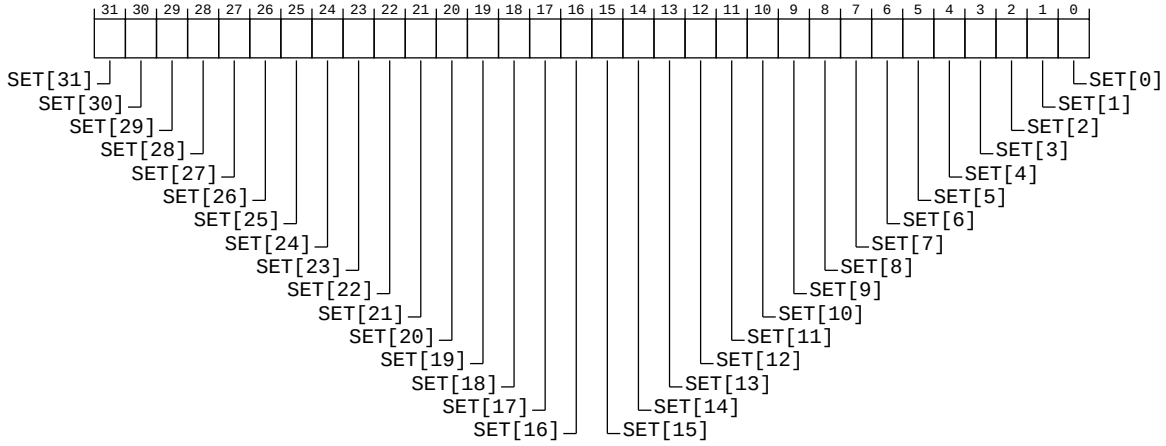


Table B-545: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:0]	SET[<m>], bit[m], where m = 31 to 0	Claim Tag Set. Indicates whether Claim Tag bit <m> is implemented, and is used to set Claim Tag bit <m> to 1. **0b0** On a read: Claim Tag bit <m> is not implemented. On a write: Ignored. **0b1** On a read: Claim Tag bit <m> is implemented. On a write: Set Claim Tag bit <m> to 1. This bit reads-as-zero and ignores writes if m > the number of Claim Tag bits.	0xFFFFFFFF

Accessibility

Component	Offset	Instance	Range
ETE	0xFA0	TRCCLAIMSET	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.54 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

Access type

RW1C

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-291: ext_trcclaimclr bit assignments

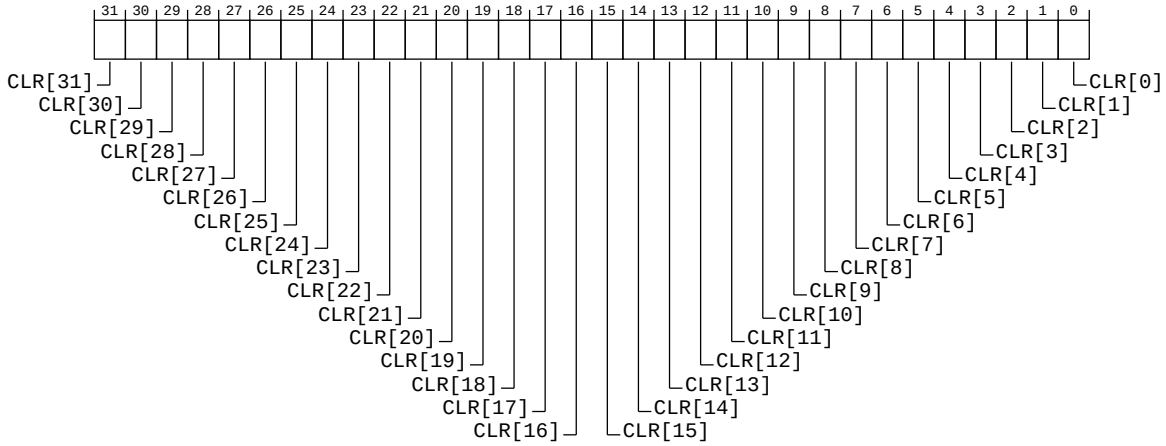


Table B-547: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:0]	CLR[<m>], bit[m], where m = 31 to 0	Claim Tag Clear. Indicates the current status of Claim Tag bit <m>, and is used to clear Claim Tag bit <m> to 0. **0b0** On a read: Claim Tag bit <m> is not set. On a write: Ignored. **0b1** On a read: Claim Tag bit <m> is set. On a write: Clear Claim tag bit <m> to 0. The number of Claim Tag bits implemented is indicated in ext-TRCCLAIMSET. This bit reads-as-zero and ignores writes if m > the number of Claim Tag bits.	0x00000000

Accessibility

Component	Offset	Instance	Range
ETE	0xFA4	TRCCLAIMCLR	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.55 TRCDEVAFF, Device Affinity Register

For additional information, see the CoreSight Architecture Specification.

Reads the same value as the AArch64-MPIDR_EL1 register for the PE that this trace unit has affinity with.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

ETE

Register offset

0xFA8

Access type

See bit descriptions

Reset value

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

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-292: ext_trcdevaff bit assignments

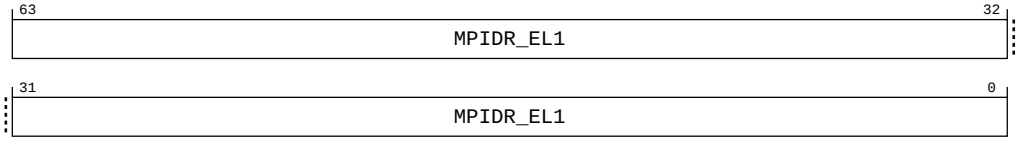


Table B-549: TRCDEVAFF bit descriptions

Bits	Name	Description	Reset
[63:0]	MPIDR_EL1	Read-only copy of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	`64{x}`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFA8	TRCDEVAFF	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.56 TRCLAR, Lock Access Register

Used to lock and unlock the Software Lock.

Note that ETE does not implement the Software Lock.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFB0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-293: ext_trclar bit assignments

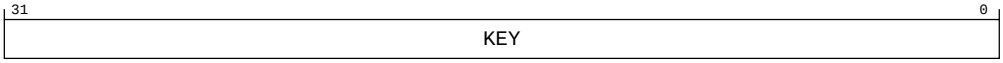


Table B-551: TRCLAR bit descriptions

Bits	Name	Description	Reset
[31:0]	KEY	Software Lock Key. A value of `0xC5ACCE55` unlocks the Software Lock. Any other value locks the Software Lock.	`32{x}`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFB0	TRCLAR	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

WO

B.7.57 TRCLSR, Lock Status Register

Indicates whether the Software Lock is implemented, and the current status of the Software Lock.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFB4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x0xx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-294: ext_trclsr bit assignments

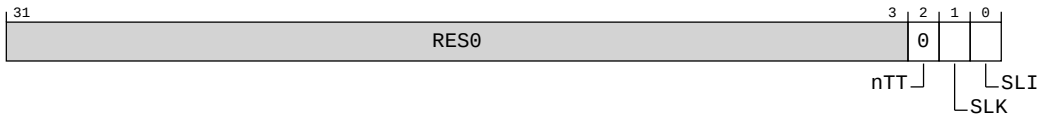


Table B-553: TRCLSR bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2]	nTT	Software lock size. **0b0**	0b0
[1]	SLK	The current Software Lock status. **0b0** Software Lock is unlocked. **0b1** Software Lock is locked. Writes to the other registers in this component, except for the ext-TRCLAR, are ignored. This field reads as 0.	x
[0]	SLI	Indicates whether the Software Lock is implemented. **0b0** Software Lock is not implemented. Writes to the ext-TRCLAR are ignored. **0b1** Software Lock is implemented. This field reads as 0.	x

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFB4	TRCLSR	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.58 TRCAUTHSTATUS, Authentication Status Register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for debug.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

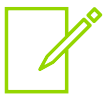
0xFB8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-295: ext_trcauthstatus bit assignments

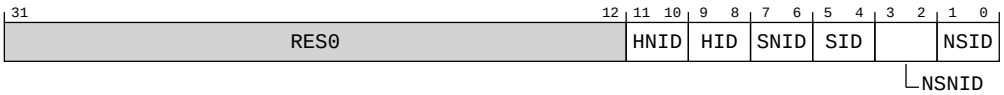


Table B-555: TRCAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:12]	**RES0**	Reserved	**RES0**
[11:10]	HNID	Hyp Non-invasive Debug. Indicates whether a separate enable control for EL2 non-invasive debug features is implemented and enabled. **0b00** Separate Hyp non-invasive debug enable not implemented, or EL2 non-invasive debug features not implemented. **0b10** Implemented and disabled. **0b11** Implemented and enabled. All other values are reserved. This field reads as 0b00.	xx
[9:8]	HID	Hyp Invasive Debug. Indicates whether a separate enable control for EL2 invasive debug features is implemented and enabled. **0b00** Separate Hyp invasive debug enable not implemented, or EL2 invasive debug features not implemented. **0b10** Implemented and disabled. **0b11** Implemented and enabled. All other values are reserved. This field reads as 0b00.	xx

Bits	Name	Description	Reset
[7:6]	SNID	Secure Non-invasive Debug. Indicates whether Secure non-invasive debug features are implemented and enabled. **0b00** Secure non-invasive debug features not implemented. **0b10** Implemented and disabled. **0b11** Implemented and enabled. All other values are reserved. When EL3 is implemented, this field takes the value `0b10` or `0b11` depending whether Secure non-invasive debug is enabled.	`xx`
[5:4]	SID	Secure Invasive Debug. Indicates whether Secure invasive debug features are implemented and enabled. **0b00** Secure invasive debug features not implemented. **0b10** Implemented and disabled. **0b11** Implemented and enabled. All other values are reserved. This field reads as `0b00`.	`xx`
[3:2]	NSNID	Non-secure Non-invasive Debug. Indicates whether Non-secure non-invasive debug features are implemented and enabled. **0b00** Non-secure non-invasive debug features not implemented. **0b10** Implemented and disabled. **0b11** Implemented and enabled. All other values are reserved. When EL3 is implemented, this field reads as `0b11`.	`xx`

Bits	Name	Description	Reset
[1:0]	NSID	Non-secure Invasive Debug. Indicates whether Non-secure invasive debug features are implemented and enabled. **0b00** Non-secure invasive debug features not implemented. **0b10** Implemented and disabled. **0b11** Implemented and enabled. All other values are reserved. This field reads as 0b00.	xx

Access

For implementations that support multiple access mechanisms, different access mechanisms can return different values for reads of TRCAUTHSTATUS if the authentication signals have changed and that change has not yet been synchronized by a Context synchronization event. This scenario can happen if, for example, the external debugger view is implemented separately from the system instruction view to allow for separate power domains, and so observes changes on the signals differently.

External debugger accesses to this register are unaffected by the OS Lock.

Accessibility

For implementations that support multiple access mechanisms, different access mechanisms can return different values for reads of TRCAUTHSTATUS if the authentication signals have changed and that change has not yet been synchronized by a Context synchronization event. This scenario can happen if, for example, the external debugger view is implemented separately from the system instruction view to allow for separate power domains, and so observes changes on the signals differently.

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFB8	TRCAUTHSTATUS	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.59 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

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-296: ext_trcdevarch bit assignments

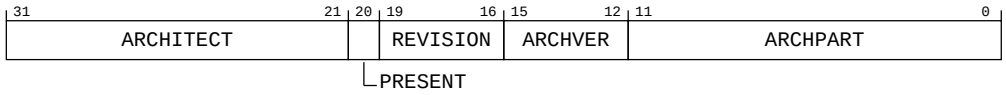


Table B-557: TRCDEVARCH bit descriptions

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.	11{x}
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **0b1** Device Architecture information present.	x

Bits	Name	Description	Reset
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0b0000** ETEv1.0, FEAT_ETE.	xxxx
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0101** ETEv1. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as 0x5.	xxxx
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0b101000010011** Arm PE trace architecture.	12{x}

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFBC	TRCDEVARCH	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.60 TRCDEVID2, Device Configuration Register 2

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFC0

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-297: ext_trcdevid2 bit assignments

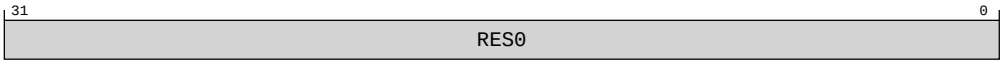


Table B-559: TRCDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC0	TRCDEVID2	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.61 TRCDEVID1, Device Configuration Register 1

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC4

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-298: ext_trcdevid1 bit assignments

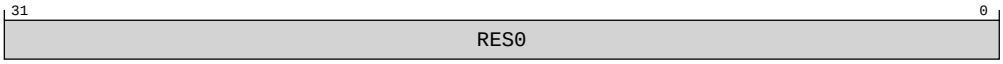


Table B-561: TRCDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC4	TRCDEVID1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.62 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

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-299: ext_trcdevid bit assignments

Table B-563: TRCDEVID bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFC8	TRCDEVID	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.63 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFCC

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-300: ext_trcdevtype bit assignments

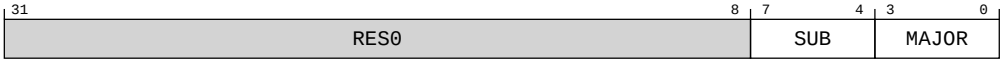


Table B-565: TRCDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Component sub-type. **0b0001** When MAJOR == 0x3 (Trace source): Associated with a PE. This field reads as 0x1.	xxxx
[3:0]	MAJOR	Component major type. **0b0011** Trace source. Other values are defined by the CoreSight Architecture. This field reads as 0x3.	xxxx

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFCC	TRCDEVTYPE	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.64 TRCPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFD0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-301: ext_trcpidr4 bit assignments

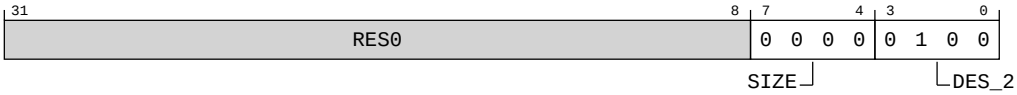


Table B-567: TRCPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. The distance from the start of the address space used by this component to the end of the component identification registers. A value of `0b0000` means one of the following is true: • The component uses a single 4KB block. • The component uses an **IMPLEMENTATION DEFINED** number of 4KB blocks. Any other value means the component occupies $2^{TRCPIDR4.SIZE}$ 4KB blocks. **`0b0000`** Using this field to indicate the size of the component is deprecated. This field might not correctly indicate the size of the component. Arm recommends that software determine the size of the component from the Unique Component Identifier fields, and other **IMPLEMENTATION DEFINED** registers in the component.	`0b0000`
[3:0]	DES_2	Designer, JEP106 continuation code. This is the JEDEC-assigned JEP106 bank identifier for the designer of the component, minus 1. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **`0b0100`** Arm Limited. This is bits[3:0] of the JEP106 continuation code. **Note:** For a component designed by Arm Limited, the JEP106 bank is 5, meaning this field has the value `0x4`.	`0b0100`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD0	TRCPIDR4	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.65 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-302: ext_trcpidr5 bit assignments

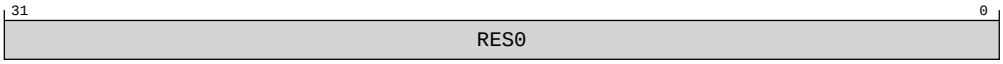


Table B-569: TRCPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD4	TRCPIDR5	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.66 TRCPIDR6, Peripheral Identification Register 6

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD8

Access type

See bit descriptions

Reset value

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-303: ext_trcpidr6 bit assignments

Table B-571: TRCPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFD8	TRCPIDR6	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.67 TRCPIDR7, Peripheral Identification Register 7

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFDC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-304: ext_trcpidr7 bit assignments

Table B-573: TRCPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFDC	TRCPIDR7	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.68 TRCPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0100 1110

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-305: ext_trcpidr0 bit assignments

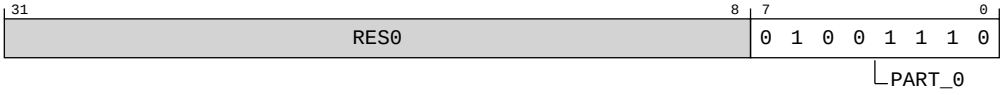


Table B-575: TRCPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, bits [7:0]. The part number is selected by the designer of the component, and is stored in ext-TRCPIDR1.PART_1 and TRCPIDR0.PART_0. **0b01001110** Least significant byte of the ETM trace unit part.	0x4E

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE0	TRCPIDR0	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.69 TRCPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-306: ext_trcpidr1 bit assignments

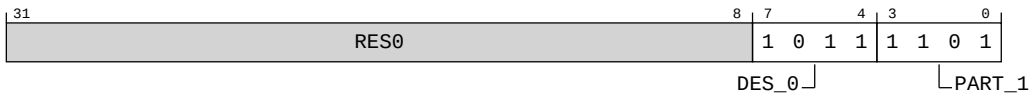


Table B-577: TRCPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	DES_0	Designer, JEP106 identification code, bits [3:0]. TRCPIDR1.DES_0 and ext-TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **0b1011** Arm Limited. This is the least significant nibble of JEP106 ID code. **Note:** For a component designed by Arm Limited, the JEP106 identification code is `0x3B`.	`0b1011`
[3:0]	PART_1	Part number, bits [11:8]. The part number is selected by the designer of the component, and is stored in TRCPIDR1.PART_1 and ext-TRCPIDR0.PART_0. **0b1101** Part number, most significant nibble.	`0b1101`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE4	TRCPIDR1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.70 TRCPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0001 1011

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-307: ext_trcpidr2 bit assignments

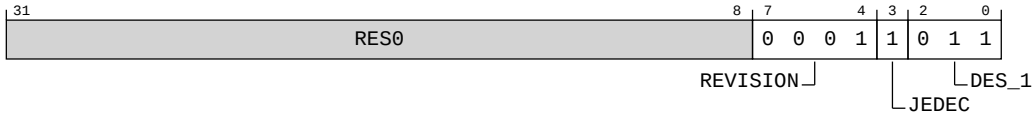


Table B-579: TRCPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component major revision. TRCPIDR2.REVISION and ext-TRCPIDR3.REVAND together form the revision number of the component, with TRCPIDR2.REVISION being the most significant part and ext-TRCPIDR3.REVAND the least significant part. When a component is changed, TRCPIDR2.REVISION or ext-TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. ext-TRCPIDR3.REVAND should be set to `0b0000` when TRCPIDR2.REVISION is increased. **`0b0001`** r1p2 - Part major revision.	`0b0001`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **`0b1`**	`0b1`

Bits	Name	Description	Reset
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. ext-TRCPIDR1.DES_0 and TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **0b011** Arm Limited. Most significant nibble of JEP106 ID code. **Note:** For a component designed by Arm Limited, the JEP106 identification code is `0x3B`.	`0b011`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFE8	TRCPIDR2	None

This interface is accessible as follows:

When !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.71 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFEC

Access type
See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0010 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-308: ext_trcpidr3 bit assignments

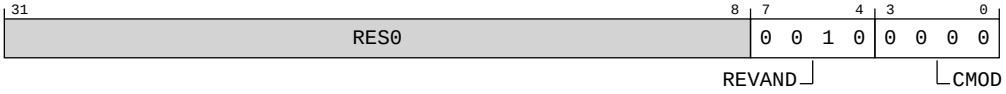


Table B-581: TRCPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Component minor revision. ext-TRCPIDR2.REVISION and TRCPIDR3.REVAND together form the revision number of the component, with ext-TRCPIDR2.REVISION being the most significant part and TRCPIDR3.REVAND the least significant part. When a component is changed, ext-TRCPIDR2.REVISION or TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. TRCPIDR3.REVAND should be set to `0b0000` when ext-TRCPIDR2.REVISION is increased. **`0b0010`** Part minor revision.	`0b0010`
[3:0]	CMOD	Customer Modified. Indicates the component has been modified. A value of `0b0000` means the component is not modified from the original design. Any other value means the component has been modified in an **IMPLEMENTATION DEFINED** way. **`0b0000`** Not Customer modified. For any two components with the same Unique Component Identifier: • If the value of the CMOD fields of both components equals zero, the components are identical. • If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have the same modifications. • If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same Unique Component Identifier.	`0b0000`

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFEC	TRCPIDR3	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.72 TRCCIDR0, Component Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF0

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-309: ext_trccidr0 bit assignments

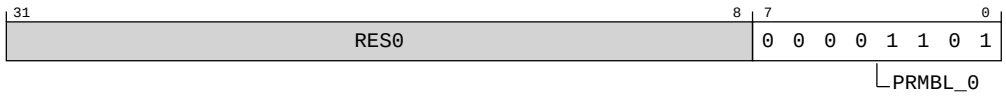


Table B-583: TRCCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Component identification preamble, segment 0. **0b00001101**	0x0D

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF0	TRCCIDR0	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.73 TRCCIDR1, Component Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF4

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-310: ext_trccidr1 bit assignments

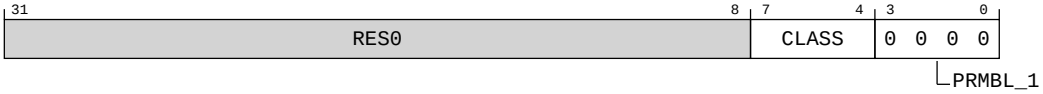


Table B-585: TRCCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	Component class. **0b1001** CoreSight peripheral.	xxxx
[3:0]	PRMBL_1	Component identification preamble, segment 1. **0b0000**	0b0000

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF4	TRCCIDR1	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.74 TRCCIDR2, Component Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF8

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-311: ext_trccidr2 bit assignments

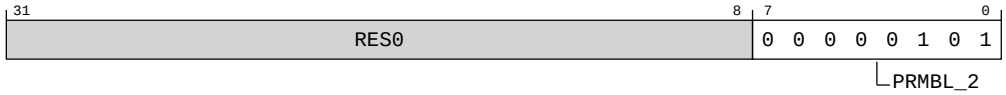


Table B-587: TRCCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	Component identification preamble, segment 2. **0b00000101**	0x05

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFF8	TRCCIDR2	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.75 TRCCIDR3, Component Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFFC

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure B-312: ext_trccidr3 bit assignments

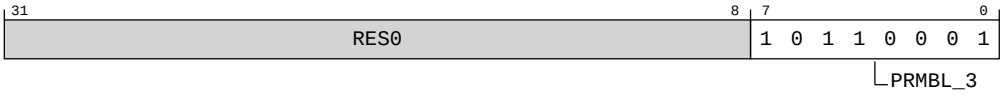


Table B-589: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Component identification preamble, segment 3. **0b10110001**	0xB1

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFFC	TRCCIDR3	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

Appendix C Document revisions

This appendix records the changes between released issues of this document.

C.1 Revisions

Changes between released issues of this book are summarized in tables.

The first table is for the first release. Then, each table compares the new issue of the book with the last released issue of the book. Release numbers match the revision history in Release Information on page 2.

Table C-1: Issue 0000-01

Change	Location
First alpha release for r0p0	-

Table C-2: Differences between issue 0000-01 and issue 0000-03

Change	Location
First Beta release for r0p0	Revision history
Added new core implementation options	2.2 Cortex-X3 core configuration options on page 30
Updated tables in Direct access to internal memory chapter	• 10.1 L1 cache encodings on page 75 • 10.2 L2 cache encodings on page 84
Updated CoreSight component identification table	16.6 CoreSight component identification on page 109
Added Statistical profiling extension support chapter	21. Statistical Profiling Extension support on page 148
Added Statisitical profiling extension registers	• A.14.3 PMBIDR_EL1, Profiling Buffer ID Register on page 957 • A.14.1 PMSEVFR_EL1, Sampling Event Filter Register on page 944 • A.14.2 PMSIDR_EL1, Sampling Profiling ID Register on page 955

Table C-3: Differences between issue 0000-03 and issue 0000-04

Change	Location
First LAC release for r0p0	Revision history
Added information on SPE and AMU	3.1 Core components on page 37
Editorial changes	6.3 Translation Lookaside Buffer match process on page 59
Updated L1 data TLB format for Data Register 2	10.1.7 L1 data TLB returned data on page 82
Updated RAM cache protection table	11.1 Cache protection behavior on page 92
Combined AArch64 RAS register summary and External register summary tables	A. AArch64 registers on page 152

Change	Location
Updates to the following AArch64 registers: • IMP_CPUECTLR2_EL1, CPU Extended Control Register 2 • ID_AA64ZFR0_EL1, SVE Feature ID register 0 • TRCIDR1, ID Register 1	A. AArch64 registers on page 152
Added the following registers: • ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0 • ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1	A. AArch64 registers on page 152
Updates to the following External registers: • PMPCSSR, Snapshot Program Counter Sample Register • EDRCR, External Debug Reserve Control Register • EDPRCR, External Debug Power/Reset Control Register • EDPFR, External Debug Processor Feature Register • TRCIDR0, ID Register 0 • TRCIDR1, ID Register 1 • TRCIDR2, ID Register 2 • TRCIDR3, ID Register 3	B. External registers on page 960
Removed PMSSRR, PMU Snapshot Reset Register	B. External registers on page 960

Table C-4: Differences between issue 0000-04 and issue 0100-05

Change	Location
First EAC release for r1p0	Revision history
Updated Feature names to allign with *Arm® Architecture Reference Manual for A-profile architecture*	2.4 Supported standards and specifications on page 31
Added paragraph to External aborts	6.6 Responses on page 61
Editorial changes to tables	10.1.1 L1 instruction tag RAM returned data on page 77
Updated bits [53:20] in L2 TLB format for Instruction Register 0 table	10.2.3 L2 TLB RAM returned data on page 88
Modified introduction to Debug chapter	• 16. Debug on page 104 • 16.1 Supported debug methods on page 105
Updated table to reflect r1p0 changes	16.6 CoreSight component identification on page 109
Corrrected PMU_HOVFS event mnemonic	18.7 ETE events on page 135
Minor changes to [45:44], [43], and [42] bits in IMP_CPUECTLR_EL1 bit descriptions table	A.1.10 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 173
Minor changes to [29] DIC bit in CTR_EL0 bit descriptions table	A.4.22 CTR_EL0, Cache Type Register on page 401
Corrected bit values	A.13.6 ERXADDR_EL1, Selected Error Record Address Register on page 916
Updated ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register	A.13.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register on page 923
Bit values set to zero	B.1.5 COREROM_AUTHSTATUS, Core ROM table Authentication Status Register on page 966

Table C-5: Differences between issue 0100-05 and issue 0101-06

Change	Location
First release for r1p1	Revision history
Added description on FEAT_ECBHB	2.4 Supported standards and specifications on page 31
Added Performance and power management	5.5 Performance and power management on page 52
Added Cortex®-X3 core powerup and powerdown sequence	5.6 Cortex-X3 core powerup and powerdown sequence on page 54
Added Write streaming mode	8.5 Write streaming mode on page 71
Update to Activity monitors events	20.3 Activity monitors events on page 144
Updated Table 16-3 Coresight component identification	16.6 CoreSight component identification on page 109

Table C-6: Differences between issue 0101-06 and issue 0102-07

Change	Location
First Non-Confidential release for r1p2	-
Changes to system ID registers throughout document	-