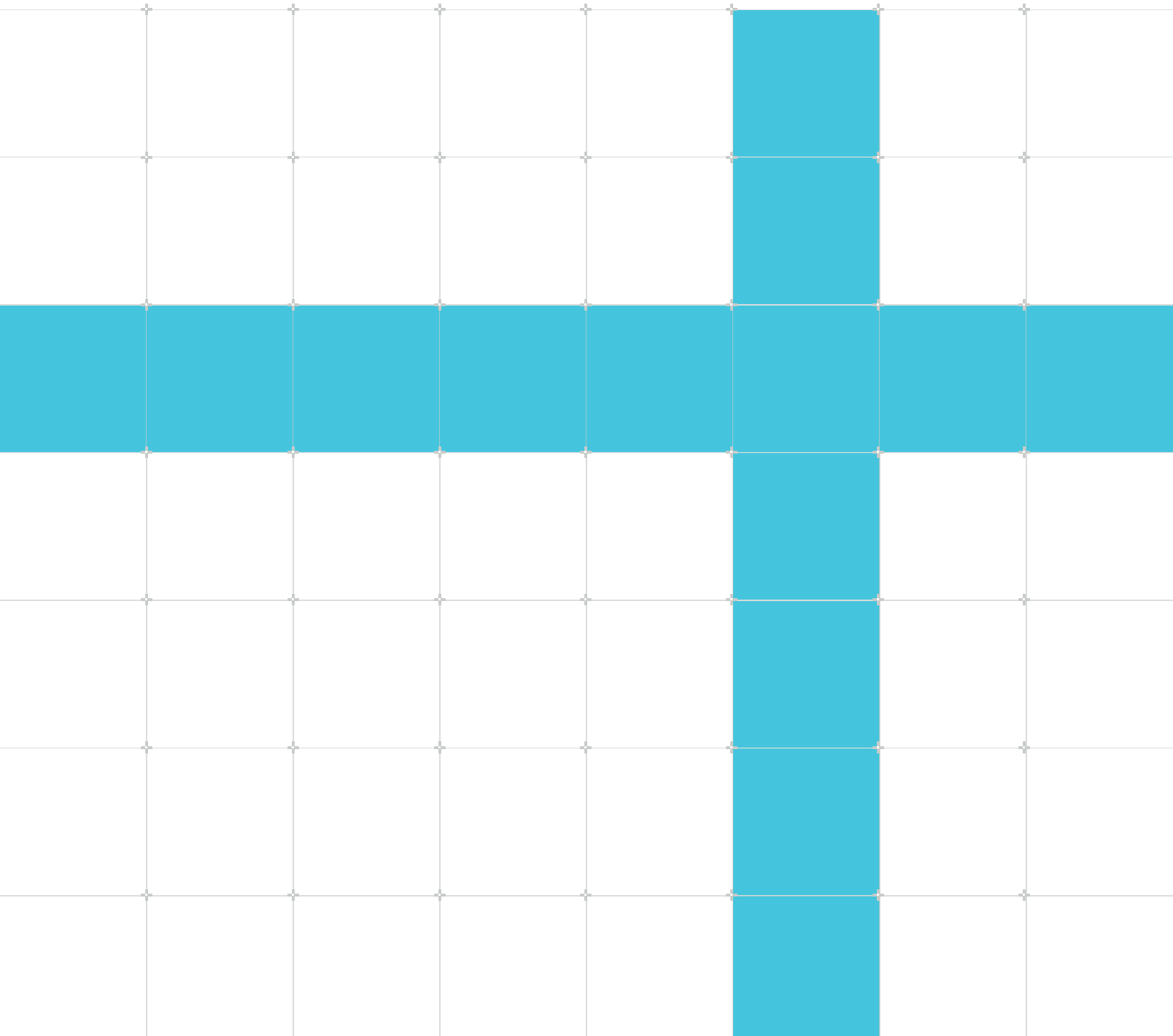
Arm® Neoverse™ V3 Core

Revision: r0p1

Technical Reference Manual

Non-Confidential

Issue 04
107734_0001_04_en



Arm® Neoverse™ V3 Core

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-02	27 March 2023	Confidential	First early access release for r0p0
0001-03	1 November 2023	Confidential	First release for r0p1
0001-04	21 February 2024	Non-Confidential	Second early access release for r0p1

Proprietary Notice

REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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

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

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Feedback

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

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

Inclusive language commitment

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

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

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

Contents

1. Introduction

1.1 Product revision status

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

r*x* Identifies the major revision of the product, for example, r1.
p*y* Identifies the minor revision or modification status of the product, for example, p2.

1.2 Intended audience

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

1.3 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

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

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

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

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

We recommend the following. If you do not follow these recommendations your system might not work.

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or of harming yourself.

This is important information and needs your attention.

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

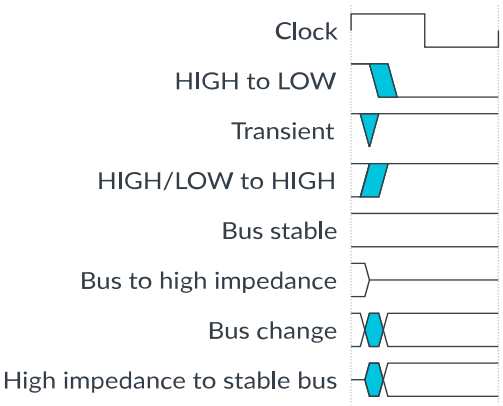
A reminder of something important that relates to the information you are reading.

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®DynamIQ™ Shared Unit-120 Configuration and Integration Manual	102548	Confidential
Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual	102547	Confidential
Arm® Neoverse™ V3 Core Configuration and Integration Manual	107735	Confidential
Arm® Neoverse™ V3 Core Cryptographic Extension Technical Reference Manual	107736	Non-Confidential
Neoverse™ V3 Release Note	-	Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® 5 CHI Architecture Specification	IHI 0050	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential
Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture	DDI 0598	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® Architecture Reference Manual Supplement, Reliability, Availability, and Serviceability (RAS), for A-profile architecture	DDI 0587	Non-Confidential


Note

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

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

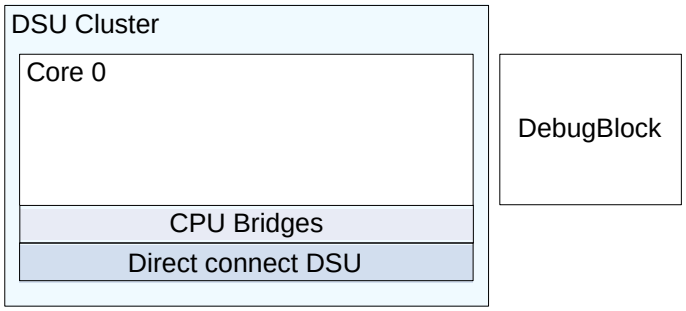
2. The Neoverse™ V3 core

The Neoverse™ V3 core is a high-performance and low-power product that implements the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A.

The Neoverse™ V3 core is implemented inside a DSU-120 cluster and is always connected to the DSU-120. The Neoverse™ V3 core supports Direct connect only. For more information on the DSU Direct connect, see the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.

The following figure shows an example configuration with one Neoverse™ V3 core that is implemented as a single core in a DSU cluster which is configured for Direct connect, without the L3 cache, snoop filter, or *Snoop Control Unit* (SCU) logic present.

Figure 2-1: Neoverse™ V3 example configuration



Note

- This manual applies to the Neoverse™ V3 core only. Read this manual together with the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for detailed information about the DSU-120.
- This manual does not provide a complete list of registers. Read this manual together with the *Arm® Architecture Reference Manual for A-profile architecture*.

2.1 Neoverse™ V3 core features

The Neoverse™ V3 core is used in a standalone DSU configuration, which is configured in Direct connect mode.

Regardless of the cluster configuration, the Neoverse™ V3 core always has the same features as described in the following lists.

Core features

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

The Cryptographic Extension is licensed separately.

Cache features

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

Debug features

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


Note

The ELA-600 is licensed separately.

Related information

3. Technical overview on page 40

2.2 Neoverse™ V3 core configuration options

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

The Neoverse™ V3 core configuration options include:

Vector datapath
: You can configure the Vector datapath to be 2x128 bits or 4x128 bits.

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

Coherent instruction cache
: You can configure your implementation with or without support for coherent instruction cache.

Random Number Generator
: You can configure your implementation with or without support for Armv8.5-RNG.

L2 cache size
: You can configure the L2 cache to be 2MB or 3MB.

CoreSight™ Embedded Logic Analyzer (ELA)
: You can include support for integrating ELA-600 as a separate licensable product.

Size of the ATB FIFO depth in the core ELA
: You can configure the size of the 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 the *poseidonv.yaml file* section in the *RTL configuration process* chapter of the *Arm® Neoverse™ V3 Core Configuration and Integration Manual*.

Register File Parity
: You can configure your implementation with or without Register File Parity.

For detailed configuration options and guidelines, see *RTL configuration process* in the *Arm® Neoverse™ V3 Core Configuration and Integration Manual*.

2.3 DSU-120 dependent features

Some DSU-120 features and behaviors depend on whether your licensed core supports a particular feature.

The following table describes which DSU-120 dependent features are supported in your Neoverse™ V3 core.

Table 2-1: Neoverse™ V3 core features that have a dependency on the DSU-120

Feature	Supported in the Neoverse™ V3 core	Dependency on the DSU-120
Direct connect	Only supports Direct connect	Direct connect support at the DSU-120 cluster level only applies when your licensed core also supports Direct connect.
Core included in a complex	No	Affects the DSU-120 cluster configuration and external signals.
Cryptographic Extension	Yes, as an option	Affects the external signals of the DSU-120. For more information on MPMM, PDP, and the dispatch block signal, see 5.5 Performance and power management on page 55.
Maximum Power Mitigation Mechanism (MPMM)	Yes	
Performance Defined Power (PDP) feature	Yes	
DISPBLKy Dispatch block signal	Yes	
Statistical Profiling Extension (SPE) architecture	Yes	
Physical Address (PA) width	48-bit	Affects the CHI master and AXI master port bus widths. For more details, see the following chapters of the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*: • *CHI master interface* • *AXI master interface*

Note

- The Cryptographic Extension is supplied under a separate license.
- The SMCRYPTODISABLE is a DSU-120 dependent feature supported in the Neoverse™ V3 core. See the *DynamIQ™ Shared Unit-120 signals* section in the *Functional integration* chapter of the *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* for the connection information of this signal.

2.4 Supported standards and specifications

The Neoverse™ V3 core complies with the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A. The Neoverse™ V3 core also complies with specific Arm®v8-A architecture extensions and supports interconnect, interrupt, timer, debug, and trace architectures.

The Neoverse™ V3 core supports AArch64 only at Exception levels EL0 to EL3.

Not all architectural features are implemented in the Neoverse™ V3 core. The following tables show the implementation status of Arm®v8-A and Arm®v9-A features supported by the Neoverse™ V3 core. There is a separate table for each version of the Arm®v8-A and Arm®v9-A architectures.


Note

- Not all Arm®v8-A and Arm®v9-A architectural features are listed in the following tables. For more information on all the architectural features, see the *Arm® Architecture Reference Manual for A-profile architecture*.
- The Neoverse™ V3 core is compatible with the architecture for the DSU-120. See the *Supported standards and specifications* section in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for a list of specific architectural versions and features supported by the DSU-120.

Table 2-2: Implementation status of the Arm®v8.0-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_PCSRv8	No	PC Sample-based Profiling Extension
Cryptographic Extension	Yes Configurable	For more information and additional cryptographic register descriptions, see the *Arm® Neoverse™ V3 Core Cryptographic Extension Technical Reference Manual*. This extension is licensed separately and access to the documentation is restricted by contract with Arm.
FEAT_SHA1	Yes Configurable Supported as part of the Arm®v8-A Cryptographic Extension	Advanced SIMD SHA1 instructions
FEAT_SHA256		Advanced SIMD SHA256 instructions
FEAT_AES		Advanced SIMD AES instructions
FEAT_PMULL		Advanced SIMD PMULL instructions
FEAT_DoubleLock	No	Double Lock
FEAT_CP15SDISABLE2	No	CP15DISABLE2
FEAT_FP	Yes	Floating point extension
FEAT_AdvSIMD	Yes	Advanced SIMD Extension For more information and register descriptions, see 14. Advanced SIMD and floating-point support on page 109.
FEAT_CRC32	Yes	CRC32 instructions
FEAT_PMUv3	Yes	PMU extension version 3
FEAT_nTLBPA	Yes	No intermediate caching by output address in TLB
FEAT_SB	Yes	Speculation barrier
FEAT_SSBS	Yes	Speculative Store Bypass Safe Instruction
FEAT_SSBS2	Yes	MRS and MSR instructions for SSBS version 2
FEAT_CSV2	Yes	Cache Speculation Variant 2
FEAT_CSV2_1p1	No	Cache Speculation Variant 2 version 1.1

Feature	Implemented	Description
FEAT_CSV2_1p2	No	Cache Speculation Variant 2 version 1.2
FEAT_CSV2_2	Yes	Cache Speculation Variant 2 version 2
FEAT_CSV3	Yes	Cache Speculation Variant 3
FEAT_SPECRES	Yes	Speculation restriction instructions
FEAT_DGH	Yes	Data Gathering Hint
FEAT_ETS	Yes	Enhanced Translation Synchronization
FEAT_ECBHB	Yes	*Exploitative Control using Branch History Buffer* information between exception levels The branch history information created in a context before an exception to a higher exception level, using AArch64, cannot be used by code before that exception. This prevents exploitative control of the execution of any indirect branches in code in a different context after the exception.

Table 2-3: Implementation status of the Arm®v8.1-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_LSE	Yes	Large System Extensions
FEAT_RDM	Yes	Rounding double multiply accumulate
FEAT_HPDS	Yes	Hierarchical permission disables in translation tables
FEAT_VHE	Yes	Virtualization Host Extensions
FEAT_PAN	Yes	Privileged access-never
FEAT_LOR	Yes	Limited ordering regions
FEAT_HAFDBS	Yes	Hardware updates to access flag and dirty state in translation tables
FEAT_VMID16	Yes	16-bit VMID
FEAT_PMUv3p1	Yes	PMU extensions version 3.1
FEAT_Debugv8p1	Yes	Debug with VHE
FEAT_PAN3	Yes	Support for SCTLR_ELx.EPAN

Table 2-4: Implementation status of the Arm®v8.2-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_TTCNP	Yes	Common not private translations
FEAT_XNX	Yes	Execute-never control distinction by Exception level at stage 2
FEAT_UAO	Yes	Unprivileged Access Override control
FEAT_PAN2	Yes	AT S1E1R and AT S1E1W instruction variants for PAN
FEAT_DPB	Yes	DC CVAP instruction
FEAT_Debugv8p2	Yes	Arm®v8.2-A Debug
FEAT_IESB	Yes	Implicit Error synchronization event
FEAT_AA32HPD	No	AArch32 Hierarchical permission disables
FEAT_HPDS2	Yes	Hierarchical permission disables in translation tables 2
FEAT_LSMAOC	No	Load/Store instruction multiple atomicity and ordering controls
FEAT_FP16	Yes	Half-precision floating-point data processing
FEAT_LVA	No	Large VA support
FEAT_LPA	No	Large PA and IPA support

Feature	Implemented	Description
FEAT_VPIPT	No	VMID-aware PIPT instruction cache
FEAT_PCSRv8p2	Yes	PC Sample-based profiling version 8.2
FEAT_RAS	Yes	*Reliability, Availability, and Serviceability* (RAS) Extension version 1.1
FEAT_SPE	Yes	*Statistical Profiling Extension* (SPE) For more information, see 23. Statistical Profiling Extension support on page 179.
FEAT_SVE	Yes	*Scalable Vector Extension* (SVE)
FEAT_SHA512	Yes	Advanced SIMD SHA512 instructions
FEAT_SHA3	Configurable	Advanced SIMD EOR3, RAX1, XAR, and BCAX instructions
FEAT_SM3		Advanced SIMD SM3 instructions
FEAT_SM4	Supported as part of the Arm®v8-A Cryptographic Extension	Advanced SIMD SM4 instructions
FEAT_DotProd	Yes	Advanced SIMD Int8 dot product instructions
FEAT_FHM	Yes	Half-precision floating-point FMLAL instructions
FEAT_EVT	Yes	Enhanced Virtualization Traps
FEAT_DPB2	Yes	DC CVADP instruction
FEAT_BF16	Yes	AArch64 BFloat16 instructions
FEAT_AA32BF16	No	AArch32 BFloat16 instructions
FEAT_I8MM	Yes	Int8 Matrix Multiplication
FEAT_AA32I8MM	No	AArch32 Int8 Matrix Multiplication
FEAT_F32MM	No	SVE single-precision floating-point matrix multiply instruction
FEAT_F64MM	No	SVE double-precision floating-point matrix multiply instruction

Table 2-5: Implementation status of the Arm®v8.3-A features in the Neoverse™ V3 core

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

Feature	Implemented	Description
FEAT_FPACCOMBINE	Yes	Faulting on combined pointer authentication instructions

Table 2-6: Implementation status of the Arm®v8.4-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_SEL2	Yes	Secure EL2
FEAT_NV2	Yes	Enhanced support for nested virtualization
FEAT_S2FWB	Yes	Stage 2 forced write-back
FEAT_DIT	Yes	Data Independent Timing instructions
FEAT_IDST	Yes	ID space trap handling
FEAT_FlagM	Yes	Condition flag manipulation
FEAT_LSE2	Yes	Large System Extensions version 2
FEAT_LRCPC2	Yes	Load-acquire RCpc instructions version 2
FEAT_TLBIOS	Yes	TLB invalidate outer-shared instructions
FEAT_TLBIRANGE	Yes	TLB range invalidate range instructions
FEAT_TTL	Yes	Translation Table Level
FEAT_BBM	Yes	Translation table break before make levels
FEAT_RASv1p1	Yes	*Reliability, Availability, and Serviceability* (RAS) Extension version 1.1 All extensions up to Arm®v9.0-A at full containment capability with *Error Correcting Code* (ECC) configured. See 11. RAS Extension support on page 97 for more information on the implementation of this extension in the core.
FEAT_DoubleFault	Yes	Double Fault Extension
FEAT_Debugv8p4	Yes	Debug relaxations and extensions version 8.4
FEAT_PMUv3p4	Yes	PMU extension version 3.4
FEAT_TRF	Yes	Self hosted Trace Extensions
FEAT_TTST	Yes	Small translation tables
FEAT_AMUv1	Yes	Activity Monitors Extension
FEAT_MPAM	Yes	*Memory Partitioning and Monitoring* (MPAM) For more information on the *Memory System Resource Partitioning and Monitoring* (MPAM) Extension, see the *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for A-profile architecture*.

Table 2-7: Implementation status of the Arm®v8.5-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_FlagM2	Yes	Condition flag manipulation version 2
FEAT_FRINTTS	Yes	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions
FEAT_ExS	No	Disabling context synchronizing exception entry and exit
FEAT_GTG	Yes	Guest translation granule size
FEAT_BTI	Yes	*Branch Target Identification* (BTI)
FEAT_E0PD	Yes	Preventing EL0 access to halves of address maps

Feature	Implemented	Description
FEAT_RNG	Yes	Random number generator
FEAT_RNG_TRAP	Yes	Trapping support for RNDR and RNDRRS
FEAT_MTE	Yes	Instruction-only Memory Tagging Extension The Neoverse™ V3 core always implements the *Memory Tagging Extension* (MTE) and therefore is compliant with the CHI.E protocol. For information on CHI.E commands inferred by MTE, see the *CHI master interface* chapter in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.
FEAT_MTE2	Yes Configurable These features are enabled by setting the BROADCASTMTE pin to 1.	Full Memory Tagging Extension
FEAT_MTE3		MTE Asymmetric Fault Handling
FEAT_PMUv3p5	Yes	PMU Extension version 3.5

Table 2-8: Implementation status of the Arm®v8.6-A features in the Neoverse™ V3 core

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

Table 2-9: Implementation status of the Arm®v8.7-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_AFP	Yes	Alternate floating-point behavior
FEAT_HCX	Yes	Support for the HCRX_EL2 register
FEAT_LPA2	No	Larger physical address for 4KB and 16KB translation granules
FEAT_LS64	Yes	Support for 64 byte loads/stores without return
FEAT_LS64_V	Yes	Support for 64-byte stores with return
FEAT_LS64_ACCDATA	Yes	Support for 64-byte EL0 stores with return
FEAT_PMUv3p7	Yes	Arm®v8.7-A PMU Extensions See 19.1 Performance monitors events on page 130.
FEAT_RPRES	No	Increased precision of Reciprocal Estimate and Reciprocal Square Root Estimate
FEAT_SPEv1p2	Yes	Arm®v8.7-A SPE See 23. Statistical Profiling Extension support on page 179.

Feature	Implemented	Description
FEAT_WFxT	Yes	WFE and WFI instructions with timeout See 5.2.1 Wait for Interrupt and Wait for Event on page 48.
FEAT_XS	Yes	XS attribute

Table 2-10: Implementation status of the Arm®v8.8-A features in the Neoverse™ V3 core

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

Table 2-11: Implementation status of the Arm®v9.0-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_ETE	Yes	*Embedded Trace Extension* (ETE) See 20. Embedded Trace Extension support on page 160.
FEAT_SVE2	Yes	*Scalable Vector Extension* (SVE) version 2 See 15. Scalable Vector Extensions support on page 110.
FEAT_SVE_AES	Yes	SVE AES instructions
FEAT_SVE_PMULL128	Configurable	SVE PMULL instructions
FEAT_SVE_SHA3	Configurable	SVE SHA-3 instructions
FEAT_SVE_SM4	Supported as part of the Arm®v8-A Cryptographic Extension	SVE SM4 instructions
FEAT_SVE_BitPerm	Yes	SVE Bit Permute
FEAT_TME	No	*Transactional Memory Extension* (TME)
FEAT_TRBE	Yes	*TRace Buffer Extension* (TRBE) See 21. Trace Buffer Extension support on page 172.

Table 2-12: Implementation status of the Arm®v9.1-A features in the Neoverse™ V3 core

Feature	Implemented	Description
FEAT_ETEv1p1	Yes	Embedded Trace Extension, version 1.1

Table 2-13: Implementation status of the Arm®v9.2-A features in the Neoverse™ V3 core

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

The following table shows the other standards and specifications that the Neoverse™ V3 core supports.

Table 2-14: Other standards and specifications supported in the Neoverse™ V3 core

Standard or specification	Version	Description
FEAT_GICv4p1	GICv4.1	*Generic Interrupt Controller* (GIC) See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.
Debug	-	Arm®v9.2-A architecture implemented with Arm®v8.4-A Debug architecture support and Arm®v8.3-A Debug over powerdown support. See the *Arm® Architecture Reference Manual for A-profile architecture* for information on this architecture.
CoreSight	v3.0	See the *Arm® CoreSight™ Architecture Specification v3.0* for more information.

Related information

3.1 Core components on page 40

2.4.1 Realm management extension

The Neoverse™ V3 core can use *Realm Management Extension* (RME) when the input signal LEGACYTZEN is LOW. If the input signal LEGACYTZEN is HIGH, then Neoverse™ V3 core reverts to TrustZone security.

For more information on RME, see *Arm® Architecture Reference Manual Supplement, The Realm Management Extension (RME), for Armv9-A*.

2.5 Test features

The Neoverse™ V3 core provides test signals that enable the use of both *Automatic Test Pattern Generation* (ATPG) and *Memory Built-In Self Test* (MBIST) to test the core logic and memory arrays.

The Neoverse™ V3 core includes an ATPG test interface that provides signals to control the *Design for Test* (DFT) features of the core. To prevent problems with DFT implementation, you must carefully consider how you use these signals.

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

For the list of test signals and information on their usage, see the *Design for Test integration guidelines* chapter in the *Arm® Neoverse™ V3 Core Configuration and Integration Manual*.

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


Note

The *Arm® Neoverse™ V3 Core Configuration and Integration Manual* and *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* are confidential documents that are available with the appropriate product licenses.

2.6 Design tasks

The Neoverse™ V3 core is delivered as a synthesizable RTL description in SystemVerilog. Before you can use the Neoverse™ V3 core, you must implement, integrate, and program it.

A different party can perform each of the following tasks:

Implementation

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

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

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

Integration

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

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

Software programming

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

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

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

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

See *RTL configuration process* in the *Arm® Neoverse™ V3 Core Configuration and Integration Manual* and in the *Arm®DynamIQ™ Shared Unit-120 Configuration and Integration Manual* for implementation options.

2.7 Product revisions

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

Table 2-15: Product revisions

Revision	Notes
r0p0	First early access release
r0p1	First early access release for r0p1. Bug fixes only.

Changes in functionality that have an impact on the documentation also appear in C.1 Revisions on page 1421.

3. Technical overview

The components in the Neoverse™ V3 core are designed to make it a high-performance core.

The main blocks include:

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

The Neoverse™ V3 core interfaces with the DSU-120 through the CPU bridge.

The Neoverse™ V3 core implements the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A.

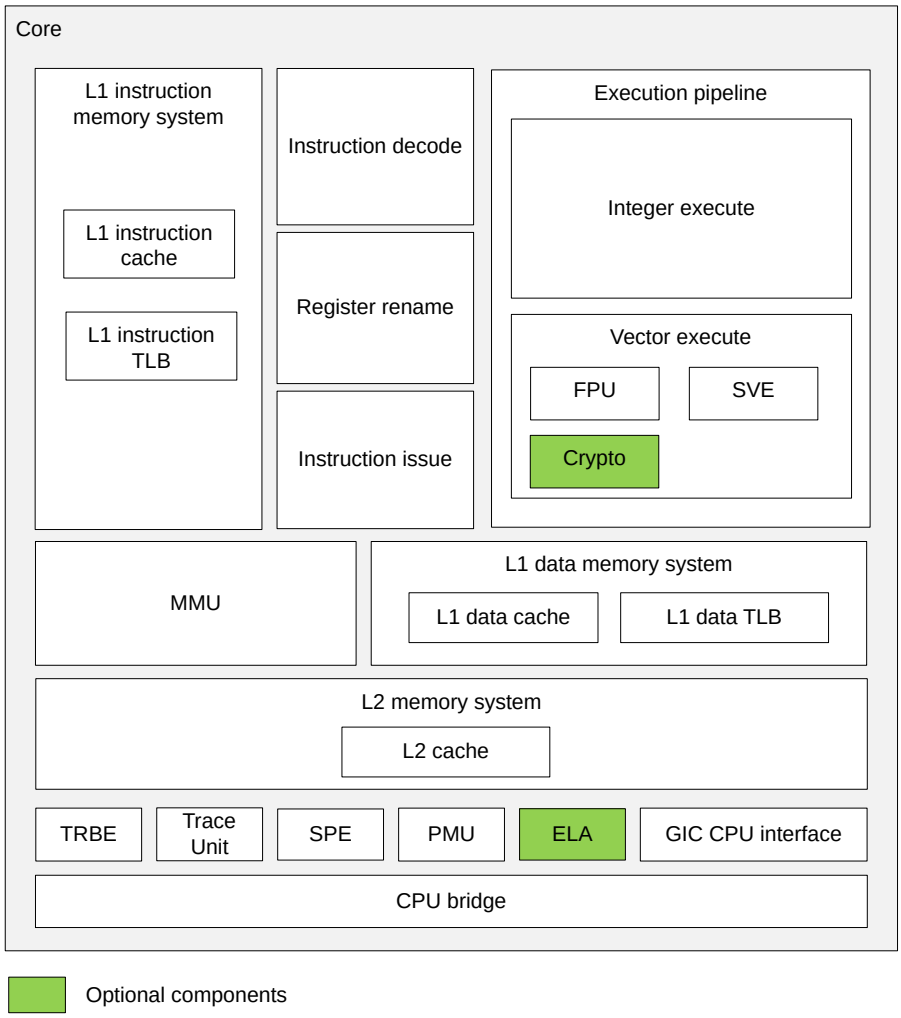
The programmer's model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in 2.4 Supported standards and specifications on page 30.

3.1 Core components

The Neoverse™ V3 core includes components designed to make it a high-performance and low-power product. The Neoverse™ V3 core includes a CPU bridge that connects the core to the DSU-120. The DSU-120 connects the core to an external memory system and the rest of the SoC.

The following figure shows the Neoverse™ V3 core components.

Figure 3-1: Neoverse™ V3 core components



L1 instruction memory system

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

The L1 instruction memory system includes:

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

Instruction decode

The instruction decode unit decodes AArch64 instructions into internal format.

Register rename

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

Instruction issue

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

Integer execute

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

Vector execute

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

Advanced SIMD and floating-point support

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

Cryptographic Extension

The Cryptographic Extension is optional in the Neoverse™ V3 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 SHA1, SHA2, SHA3.
  - The SVE2 versions of the SHA3 instructions EOR3, XAR, and BCAX are supported even when CRYPTO support is not configured.
- Armv8.2-SM SM3 hash function and SM4 encryption and decryption instructions.
- Finite field arithmetic that is used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography.


Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension under an additional license to the Neoverse™ V3 core license.

Scalable Vector Extension

The *Scalable Vector Extension* (SVE) and *Scalable Vector Extension 2* (SVE2) are extensions 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 (2MB) or 12-way (3MB) set associative. You can configure its RAM size to be 2MB or 3MB. The L2 memory system is connected to the DSU-120 through an asynchronous CPU bridge.

Embedded Trace Extension and Trace Buffer Extension

The Neoverse™ V3 core supports a range of debug, test, and trace options including a trace unit and a trace buffer.

The Neoverse™ V3 core also includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the Neoverse™ V3 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 Neoverse™ V3 core implements the *Statistical Profiling Extension* (SPE) to the Arm®v8.7-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 performance monitors. The performance monitors can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

Activity Monitoring Unit

The Neoverse™ V3 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 Neoverse™ V3 core and the DSU-120.

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

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 59
7. L1 instruction memory system on page 68
8. L1 data memory system on page 72
9. L2 memory system on page 76
13. GIC CPU interface on page 105
14. Advanced SIMD and floating-point support on page 109
19. Performance Monitors Extension support on page 130
20. Embedded Trace Extension support on page 160

3.2 Interfaces

The DSU-120 manages all Neoverse™ V3 core external interfaces to the *System on Chip* (SoC).

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

3.3 Programmer's model

The Neoverse™ V3 core implements the Arm®v9.2-A architecture. The Arm®v9.2-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.7-A. The Neoverse™ V3 core supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

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

Related information

2.4 Supported standards and specifications on page 30

4. Clocks and resets

To provide dynamic power savings, the Neoverse™ V3 core supports hierarchical clock gating. It also supports Warm and Cold resets.

The Neoverse™ V3 core has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the CPU bridge.

In addition, the Neoverse™ V3 core implements extensive clock gating that includes:

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

The Neoverse™ V3 core receives the following reset signals from the DSU-120 side of the CPU bridge:

- A Warm reset for all registers in the core except for:
  - Some parts of the Debug logic
  - Some parts of the trace unit logic
  - *Reliability, Availability, and Serviceability* (RAS) logic
  - Performance and Power Management registers. See 5.5 Performance and power management on page 55 for more details on Performance and power management registers.
- A Cold reset for the logic in the core, including the debug logic, trace logic, and RAS logic.

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

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

5. Power management

The Neoverse™ V3 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

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

The static power management includes the following features:

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

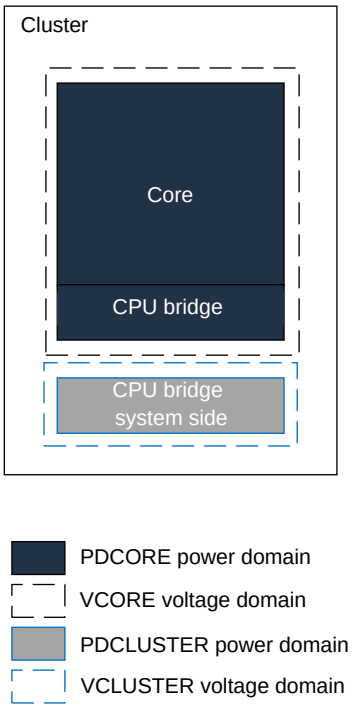
5.1 Voltage and power domains

The DSU-120 *Power Policy Units* (PPUs) control power management for the Neoverse™ V3 core. The core supports one power domain, PDCORE, and one system power domain, PDCLUSTER. Similarly, it supports one core voltage domain, VCORE, and one cluster system voltage domain, VCLUSTER. The power domains and voltage domains have the same boundaries.

The PDCORE power domain contains all Neoverse™ V3 core logic and part of the core asynchronous bridge that belongs to the VCORE domain. The PDCLUSTER power domain contains the part of the CPU bridge that belongs to the VCLUSTER domain.

The following figure shows the Neoverse™ V3 core power domain and voltage domain. It also shows the cluster power domain and voltage domain that cover the system side of the CPU bridge.

Figure 5-1: Neoverse™ V3 core voltage domains and power domains



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

- The core is configured to run synchronously with the DSU-120 sharing the same clock.
- The core is not required to support *Dynamic Voltage and Frequency Scaling* (DVFS).

Clamping cells between power domains are inferred through power intent files (UPF) rather than instantiated in the RTL. See *Power management* in the *Arm® Neoverse™ V3 Core Configuration and Integration Manual* for more information.

For detailed information on the DSU-120 cluster power domains and voltage domains, see *Power management* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*.

5.2 Architectural clock gating modes

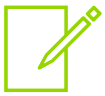
The `WFI` and `WFE` instructions put the core into a low-power mode. These instructions architecturally disable the clock at the top of the clock tree. The core remains fully powered and retains the state.

5.2.1 Wait for Interrupt and Wait for Event

Wait for Interrupt (WFI) and *Wait for Event* (WFE) are features that put 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` instructions also ensure that store instructions have updated the cache or have been issued to the system.


Note

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

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

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

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

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

5.2.2 Low-power state behavior considerations

You must consider how certain events affect the *Wait for Interrupt* (WFI) and *Wait for Event* (WFE) low-power state behavior of the Neoverse™ V3 core.

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

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


Note

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

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

5.3 Power control

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

The 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 Neoverse™ V3 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 core, see the following sections in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual*:

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

Related information

A.2.2 IMP_CPUPPMCR_EL3, Global Performance and Power Management Configuration Register on page 297
A.2.3 IMP_CPUMPMMCR_EL3, Global MPMM Control Register on page 299
B.2.1 CPUPPMCR, Global Performance and Power Management Configuration Register on page 1082
B.2.2 CPUMPMMCR, Global MPMM Control Register on page 1084

5.4 Core power modes

The Neoverse™ V3 core power domain has a defined set of power modes and corresponding legal transitions between these modes.

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

The following table shows the supported Neoverse™ V3 core power modes.

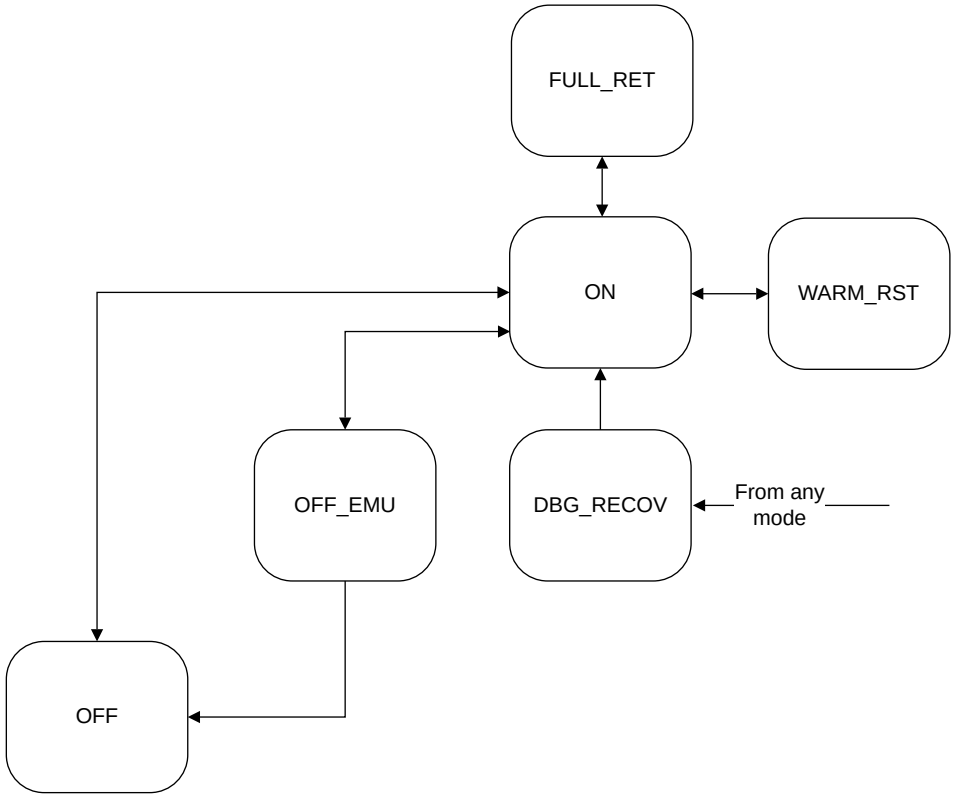
Power modes that are not shown in the following table are not supported and must not occur. Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management, and powerup and powerdown sequences described in 5.6 Neoverse V3 core powerup and powerdown sequence on page 57.

Table 5-1: Neoverse™ V3 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.
Emulated Off	OFF_EMU	Emulated off mode permits you to debug the powerup and powerdown cycle without changing the software. In this mode, the core proceeds through all the powerdown steps, except: • The clock is not gated and power is not removed when the core is powered down. • Only a Warm reset is asserted. The debug logic is preserved in the core and remains accessible by the debugger.
Debug recovery	DBG_RECOV	The RAM and logic are powered up. This mode is for applying a Warm reset to the DSU-120 cluster, while preserving memory and *Reliability, Availability, and Serviceability* (RAS) registers for debug purposes. Both cache and RAS state are preserved when transitioning from DBG_RECOV to ON. **Caution:** This mode must not be used during normal system operation.
Warm reset	WARM_RST	A Warm reset resets all state except for the trace logic and the debug and RAS registers.

The following figure shows the supported modes for the Neoverse™ V3 core power domain and the legal transitions between them.

Figure 5-2: Neoverse™ V3 core power mode transitions



Related information

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

5.4.1 On mode

In the On power mode, the Neoverse™ V3 core is on and fully operational.

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

5.4.2 Off mode

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

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

A Cold reset can reset the core in this mode.

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


Note

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

5.4.3 Emulated off mode

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

5.4.4 Full retention mode

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

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

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

- The retention timer has expired. For more information on setting the retention timer, see A.1.14 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 226.
- The core is in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state.
- The core clock is not temporarily enabled for any of the following reasons:
  - L1 snoops or L2 snoops
  - Cache or *Translation Lookaside Buffer* (TLB) maintenance operations
  - Debug or *Generic Interrupt Controller* (GIC) access

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

- A WFI or WFE wakeup event, as defined in the *Arm® Architecture Reference Manual for A-profile architecture*.
- An event that requires the core clock to be temporarily enabled without exiting the WFI or WFE low-power state. For example:
  - L1 snoops or L2 snoops
  - Cache or TLB maintenance operations
  - Debug access from the DebugBlock of the DSU-120

- GIC access

Related information

5.2.1 Wait for Interrupt and Wait for Event on page 48

5.4.5 Debug recovery mode

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

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

In addition to preserving the cache contents, Debug recovery mode supports preserving the *Reliability, Availability, and Serviceability* (RAS) state. A transition to Debug recovery mode is made from any state, which puts the core into a Warm reset state. There is no external mechanism to apply a Warm reset mode other than programming the DSU-120 *Power Policy Units* (PPUs).

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

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

Debug recovery mode can occur at any time with no guarantee of the state of the core. A request of this type is accepted immediately, therefore its effects on the core, the DSU-120 cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. In particular, any outstanding memory system transactions at the time of the reset might complete after the reset. The core is not expecting these transactions to complete after a reset, and might cause a system deadlock.

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

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

5.4.6 Warm reset mode

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

A Warm reset is applied to the Neoverse™ V3 core when the core *Power Policy Unit* (PPU) in the DSU-120 is programmed for WARM_RST mode.

Warm reset mode is only expected to be used for resets triggered by a system level issue, such as a watchdog timeout.

WARM_RST mode can occur at any time with no guarantee of the state of the core. A request to transition to WARM_RST mode is accepted immediately. Therefore, its effects on the core, the DynamIQ™ cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. For example, if there were any outstanding memory transactions at the time of the reset, these transactions might complete after the reset. Unless the system interconnect is also reset, the cluster will not expect these transactions to complete after the reset, and a system deadlock might occur.

The Neoverse™ V3 core also implements the Arm®v8-A Reset Management Register, RMR_EL3. When the core runs in EL3, it requests a Warm reset of the core if you set the RMR_EL3.RR bit to 1.

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

5.5 Performance and power management

The Neoverse™ V3 core implements *Performance and Power Management* (PPM) features that can be used to limit high activity events within the core, or trade off efficiency versus peak performance.

The PPM features are:

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

5.5.1 Maximum Power Mitigation Mechanism

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

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

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

for each gear. An external power controller can use these metrics to budget SoC power in the following ways:

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

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

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

Related information

A.2.2 IMP_CPUPPMCR_EL3, Global Performance and Power Management Configuration Register on page 297
A.2.3 IMP_CPUMPMMCR_EL3, Global MPMM Control Register on page 299
B.2.1 CPUPPMCR, Global Performance and Power Management Configuration Register on page 1082
B.2.2 CPUMPMMCR, Global MPMM Control Register on page 1084

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 core without stopping the clock.

A pin is provided on the DSU-120 boundary that can directly be used to force the core to stall for the duration that the pin is asserted. When the core is stalled, the dispatch of new instructions is

stopped. However, instructions that have already been dispatched will continue to execute and complete as normal.

5.6 Neoverse™ V3 core powerup and powerdown sequence

There is no specific sequence to power up the Neoverse™ V3 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 power down the Neoverse™ V3 core:

1. If required, save the state of the core to system memory, to allow for retrieval of the core state during power up.
2. Disable interrupts to the core.
   a. Disable the interrupt enable bits in the ICC_IGRPEN0_EL1 and ICC_IGPREN1_EL1 registers.
   b. Set the GIC distributor wake-up request for the core using the GICR_WAKER register.
   c. Read the GICR_WAKER register to confirm that the ChildrenAsleep bit indicates that the interface is inactive.
3. Optionally, disable the interrupt outputs from the RAS registers. For more information, see Managing RAS fault and error interrupts during the core powerdown procedure.
4. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
5. Execute an `ISB` instruction.
6. Execute a `WFI` instruction. Once the `WFI` instruction is executed, the powerdown sequence cannot be interrupted.

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

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

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 procedure

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

The WFI instruction is normally the point of no return for powering down the core. However. if a RAS fault or error interrupt is signaled from the core during the power down procedure, this will cause the core to deny the power down request and cause the core to wake up from WFI.

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

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

5.7 Debug over powerdown

The Neoverse™ V3 core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This behavior enables debug to continue through powerdown scenarios, rather than having to re-establish a connection each time the core is powered up.

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

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

6. Memory management

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

This translation is based on address mapping and memory attribute information that is available in the Neoverse™ V3 core internal registers and translation tables. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

An address translation from an input address to an output address is described as a stage of address translation. The Neoverse™ V3 core can perform:

- Stage 1 translations that translate an input *Virtual Address* (VA) to an output *Physical Address* (PA) or *Intermediate Physical Address* (IPA).
- Stage 2 translations that translate an input IPA to an output PA.
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The Neoverse™ V3 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 core 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.

6.1 Memory Management Unit components

The Neoverse™ V3 *Memory Management Unit* (MMU) includes several *Translation Lookaside Buffers* (TLBs), an *MMU Translation Cache* (MMUTC), and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The Neoverse™ V3 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 • 128 entries
L1 data TLB	• Caches entries at the 4KB, 16KB, 64KB, 2MB, or 512MB granularity of VA to PA mappings only • Fully associative • 96 entries
L1 *TRace Buffer Extension* (TRBE) TLB	• VA to PA translations of any page and block size • 2 entries
L2 TLB	• Shared by instructions and data • VA to PA mappings for 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, and 1GB block sizes • *Intermediate Physical Address* (IPA) to PA mappings for: ◦ 2MB and 1GB block sizes in a 4KB translation granule ◦ 32MB block size in a 16KB translation granule ◦ 512MB block size in a 64KB granule • *Intermediate PAs* (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.

6.2 Translation Lookaside Buffer entry content

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

Each TLB entry contains:

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

Each TLB entry is associated with either:

- A particular *Address Space IDentifier* (ASID)

- A global indicator

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

Related information

6.4 Translation table walks on page 62

6.3 Translation Lookaside Buffer match process

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

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

- Root EL3 or Secure EL3 when LEGACY_TZ_EN is 1'b1
- Secure EL2
- Secure EL2 and EL0
- Non-secure EL2
- Non-secure EL2 and EL0
- Realm EL2
- Realm EL2 and EL0
- Secure EL1
- Secure EL1 and EL0
- Non-secure EL1
- Non-secure EL1 and EL0
- Realm EL1
- Realm EL1 and EL0

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

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

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

- The Secure EL1 and EL0, Non-secure EL1 and EL0 and Realm EL1 and EL0 translation regime

- The Secure EL2 and EL0, Non-secure EL2 and EL0 and Realm EL2 and EL0 translation regime

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

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

6.4 Translation table walks

When the Neoverse™ V3 core generates a memory access, the *Memory Management Unit* (MMU) searches for the requested *Virtual Address* (VA) in the *Translation Lookaside Buffers* (TLBs). If it is not present, then it is a miss and the MMU proceeds by looking up the translation table during a translation table walk.

When the Neoverse™ V3 core generates a memory access, the MMU:

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

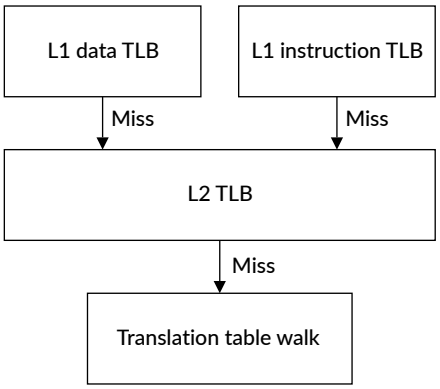
Address translation is performed only when the MMU is enabled. It can also be disabled for a particular translation base register, in which case the MMU returns a Translation Fault.

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

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

The following figure shows the TLB lookup process.

Figure 6-1: Translation table walks



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

Related information

7. L1 instruction memory system on page 68
8. L1 data memory system on page 72
9. L2 memory system on page 76

6.5 Hardware management of the Access flag and dirty state

The Neoverse™ V3 core includes the option to perform hardware updates to the translation tables.

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

The Neoverse™ V3 core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the Neoverse™ V3 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 or Root or Realm 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.

Granule Protection Table faults

When granule protection checks are enabled, faults can include the following:

- *Granule Protection Table* (GPT) address size fault
- GPT walk fault
- Granule protection 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.

Neoverse™ V3 takes a synchronous abort on a Normal memory ldrx that receives an non-EXOK response from CHI. The abort is asynchronous for Device memory ldrx. For strx, OK and EXOK responses are expected and do not cause aborts. NDErr and DErr responses for WriteNoSnp Excl=1 cause asynchronous aborts.

Misprogramming contiguous hints

When there is a descriptor that contains a set CH bit, the input *Virtual Address* (VA) address space must include all contiguous VAs contained in this block.

The VA address space is defined by:

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

The Neoverse™ V3 core treats such a block as not causing a translation fault and disregards the value of the contiguous bit.

Conflict aborts

The Neoverse™ V3 core does not generate conflict abort exceptions.

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

6.7 Memory behavior and supported memory types

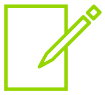
The Neoverse™ V3 core supports memory types defined in the Armv8-A architecture.

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 acknowledgement of write transactions from the interconnect


Note

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

The following table shows how memory types are supported in the Neoverse™ V3 core.

Table 6-2: Supported memory types

Memory 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

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

Memory type	Shareability	Inner Cacheability	Outer Cacheability	Notes
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 66.	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 CHI interconnect is 0 (No Allocate)
Normal	See Table 6-3: Shareability for Normal memory on page 66.	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 CHI interconnect 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	Outer Shareable
Outer Shareable	Outer Shareable
Inner Shareable	Outer Shareable

6.8 Page-based hardware attributes

The architecture defines *Page-Based Hardware Attributes* (PBHA) as an optional **IMPLEMENTATION DEFINED** feature. This section describes how the Neoverse™ V3 core implements PBHA.

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

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

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

PBHA combination between stage 1 and stage 2 on memory accesses

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

When stage 1 and stage 2 are enabled:

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

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

Mismatched aliases

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

7. L1 instruction memory system

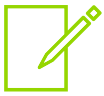
The Neoverse™ V3 core L1 instruction memory system fetches instructions and predicts branches. It includes the L1 instruction cache and the L1 instruction *Translation Lookaside Buffer* (TLB).

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

The following table shows the L1 instruction memory system features.

Table 7-1: L1 instruction memory system features

Feature	Description
L1 instruction cache	64KB
	4-way set associative
	Virtually Indexed, Physically Tagged (VIPT) behaving as *Physically Indexed, Physically Tagged* (PIPT)
	Always protected with parity
Cache line length	64 bytes
Cache policy	**L1 I-cache** Pseudo-*Least Recently Used* (LRU) cache replacement policy for L1


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 59.

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 54

7.2 L1 instruction cache Speculative memory accesses

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

On instruction fetches, pages with Device memory type attributes are treated as Non-Cacheable Normal Memory. To prevent instruction fetches, device memory pages must be marked with the translation table descriptor attribute bit *eXecute Never* (XN). The device and code address spaces must be separated in the physical memory map. This separation prevents Speculative fetches to read-sensitive devices when address translation is disabled.

If the L1 instruction cache is enabled and the instruction fetches miss in the L1 instruction cache, then they will look up in L2 and snoop the L1 data cache if those caches are enabled. Instruction fetches are always coherent with data caches, so cache maintenance operations are not required to make stores visible to instruction fetches.

However, the lookup never causes an L1 data cache refill, regardless of the data cache enable status. The line is only allocated in the L2 cache, provided that the L1 instruction cache is enabled. See the *Arm® Architecture Reference Manual for A-profile architecture* for more information.

7.3 Program flow prediction

The Neoverse™ V3 core contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and enhances power efficiency.

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

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

Program flow prediction hardware contains the following functionality:

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

Predicted and non-predicted instructions

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

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

The following instructions are not predicted:

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

Return stack

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

Any of the following instructions causes a return stack push:

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

Any of the following instructions cause a return stack pop:

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

7.4 Instruction Prefetch

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

Preload instructions

The Neoverse™ V3 core supports the AArch64 prefetch memory instructions, `PRFM PLI`, into the L1 instruction cache or L2 cache.

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

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

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

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

7.5 Instruction cache hardware coherency

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

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

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

- The coherency domain containing a core configured with instruction cache hardware coherency must not contain any coherent agents that require software instruction cache maintenance.
- Arm recommends systems consisting of a large number of Neoverse™ V3 cores should configure the cores with instruction cache coherency to eliminate possible performance issues related to instruction cache instruction broadcasts as DVMMsg transactions to all masters in the system.

8. L1 data memory system

The Neoverse™ V3 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 policy
Interface with integer execute pipeline and vector execute	• 4×64-bit read paths and 4×64-bit write paths for the integer execute pipeline • 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 59.

8.1 L1 data cache behavior

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

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

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

Data Cacheability disabled behavior

If the data Cacheability is disabled, then:

- A new line is not allocated in the L2 cache 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 Neoverse™ V3 core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

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

Related information

5.4.5 Debug recovery mode on page 54

8.2 Write streaming mode

The Neoverse™ V3 core supports write streaming mode, sometimes referred to as read allocate mode, both for the L1 and the L2 cache.

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

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

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


Note

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

The memory system continues in write streaming mode until either:

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

When a Neoverse™ V3 core has switched to write streaming mode, the memory system continues to monitor the write traffic. It signals to the L2, System Level Cache, or DRAM, 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 208

8.3 Instruction implementation in the L1 data memory system

The Neoverse™ V3 core supports the atomic instructions added in the Arm®v8.1-A architecture. Atomic instructions to Cacheable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides.

If an instruction hits in the L1 data cache, then the Neoverse™ V3 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.

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

The Neoverse™ V3 core supports atomics to Device or Non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, then it results in an abort.

8.4 Internal exclusive monitor

The Neoverse™ V3 core includes an internal exclusive monitor with a 2-state, open and exclusive state machine that manages Load-Exclusive and Store-Exclusive accesses and Clear-Exclusive (CLREX) instructions.

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


Note

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

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

See A.6.22 CTR_EL0, Cache Type Register on page 459 for more technical reference and register information.

8.5 Data prefetching

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

Hardware data prefetcher

The Neoverse™ V3 core includes multiple hardware prefetch engines as part of the L1 and L2 caches. These prefetch engines prefetch data into the L1 and L2 caches using a combination of *Virtual Address* (VA) and *Physical Address* (PA).

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

- A.1.10 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 208
- A.1.11 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2 on page 218

Data cache zero

In the Neoverse™ V3 core, the *Data Cache Zero by Virtual Address* (DC ZVA) instruction sets a 64-byte block of memory, which is aligned to 64 bytes, to zero.

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

9. L2 memory system

The Neoverse™ V3 core L2 memory system connects the core with the DSU-120 through the CPU bridge. It includes the private L2 cache.

The L2 cache is unified and private to each Neoverse™ V3 core in a cluster.

The following table shows the L2 memory system features.

Table 9-1: L2 memory system features

Feature	Description
L2 cache	• 2MB or 3MB • 8-way (2MB) or 12-way (3MB) set associative, 4 banks • *Physically Indexed, Physically Tagged* (PIPT) • Protected with *Error Correcting Code* (ECC)
Cache line length	64 bytes
Cache policy	Dynamic biased cache replacement policy
Coherent Interconnect	One CHI Issue F compliant interface with 256-bit read and data 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.

- When COHERENT_ICACHE parameter is FALSE: 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.
- When COHERENT_ICACHE parameter is TRUE: The L1 instruction cache and L2 cache are strictly inclusive. Any data contained in the L1 instruction cache is also present in the L2 cache. Victimization from the L2 cache can cause invalidations of the L1 instruction cache.

The L1 data cache and L2 cache are strictly inclusive. Any data contained in the L1 data cache is also present in the L2 cache. Victimization of L2 data can cause invalidations of the L1 data cache.

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

Related information

5.4.5 Debug recovery mode on page 54

9.2 Support for memory types

The Neoverse™ V3 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 interface between the Neoverse™ V3 core L2 memory system and the DSU-120 provides transaction capabilities for the core.

The following table shows the maximum possible values for read, write, *Distributed Virtual Memory* (DVM) issuing, and snoop capabilities of the Neoverse™ V3 core L2 cache.

Table 9-2: Neoverse™ V3 transaction capabilities

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

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

10. Direct access to internal memory

The Neoverse™ V3 core provides a mechanism to read the internal memory that the L1 caches, L2 cache, and *Translation Lookaside Buffer* (TLB) structures use through **IMPLEMENTATION DEFINED** System registers. When the coherency between the cache data and the system memory data is broken, you can use this mechanism to investigate any issues.


Note

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

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

You can access the contents of the internal memory using the six read-only (RO) System registers in Table 10-1: System registers used to access internal memory on page 78. The internal memory is selected by programing the **IMPLEMENTATION DEFINED** RAMINDEX register using the following `SYS` instruction:

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

For more information on the RAMINDEX register, see A.5.1 SYS_IMP_RAMINDEX, RAM Index on page 403. The data is read from the read-only System registers as shown in the following table.


Note

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

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

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

10.1 L1 cache encodings

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

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

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

Table 10-2: Neoverse™ V3 L1 instruction cache tag location encoding

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

Table 10-3: Neoverse™ V3 L1 instruction cache data location encoding

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

Table 10-4: Neoverse™ V3 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: Neoverse™ V3 L1 data cache tag location encoding

Bit field of Xn	Description
[31:24]	RAMID = `0x08`
[23:20]	Reserved
[19:18]	Way

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

Table 10-6: Neoverse™ V3 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]	Virtual address [13:6]
[5:0]	Reserved

Table 10-7: Neoverse™ V3 L1 data TLB location encoding

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

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 10-8: L1 instruction cache tag format for Instruction Register 0

Bit field	Description
[63:41]	Reserved
[40:39]	{NSE, NS} Non-secure identifier for the physical address
[38:3]	Physical address [47:12]

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

Table 10-9: L1 instruction cache tag format for Instruction Register 1

Bit field	Description
[63:0]	0

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

Bit field	Description
[63:0]	0

10.1.2 L1 instruction data RAM returned data

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

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

Table 10-11: L1 instruction cache data format for Instruction Register 0

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

Table 10-12: L1 instruction cache data format for Instruction Register 1

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

Table 10-13: 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 10-14: L1 instruction TLB format for Instruction Register 0

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

Bit field	Description
[40]	Reserved
[39:24]	ASID[15:0]
[23:8]	VMID[15:0]
[7:5]	MSID[2:0]: **0b000** Secure EL1/EL0 **0b001** Secure EL2 **0b010** Non-Secure EL1 **0b011** Realm EL1 **0b111** Realm EL2
[4:1]	Reserved
[0]	Valid

Table 10-15: L1 instruction TLB format for Instruction Register 1

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

Table 10-16: L1 instruction TLB format for Instruction Register 2

Bit field	Description
[63:12]	Reserved
[11]	Reserved
[10]	Reserved
[9:8]	NSE, NS (Extended Non-Secure)
[7:0]	Physical addresss[47:40]

10.1.4 L1 data tag RAM returned data

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

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

Table 10-17: L1 data cache tag format for Data Register 0

Bit field	Description
[63:28]	Physical address [47:12]
[27:25]	Reserved
[24]	Transient/WBNA

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

Table 10-18: L1 data cache tag format for Data Register 1

Bit field	Description
[63:10]	0
[9:2]	ECC
[1:0]	Physical address space `0b00` - Secure `0b01` - Non-secure `0b10` - Root `0b11` - Realm

Table 10-19: L1 data cache tag format for Data Register 2

Bit field	Description
[63:0]	0

10.1.5 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 10-20: L1 data cache data format for Data Register 0

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

Table 10-21: L1 data cache data format for Data Register 1

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

Table 10-22: L1 data cache data format for Data Register 2

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

10.1.6 L1 data TLB returned data

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

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

Table 10-23: L1 data TLB format for Data Register 0

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

Bit field	Description
[54:52]	Memory attributes [2:0]: **0b000** Device nGnRnE **0b001** Device nGnRE **0b010** Device nGRE **0b011** Device GRE **0b100** Non-cacheable **0b101** Write-Back No-Allocate **0b110** Write-Back Transient **0b111** Write-Back Read-Allocate and Write-Allocate
[51]	Outer allocate
[50]	S2 *Dirty Bit Modifier* (DBM) bit
[49]	S1 DBM bit
[48]	TLB coalesced bit
[47:44]	Permission bit [3:0]
[43]	Device/Non-cacheable HTRAP
[42]	nG bit
[41]	Smash bit
[40:38]	Page size [2:0]: **0b000** 4KB **0b001** 16KB **0b010** 64KB **0b011** 256KB **0b100** 2MB **0b101** Reserved **0b110** 512MB **0b111** Reserved

Bit field	Description
[37:36]	NSX bits[1:0] **0b00** Secure **0b01** Non Secure **0b10** Root **0b11** Realm
[35:33]	MSID[2:0]
[32:17]	ASID [15:0]
[16:1]	VMID [15:0]
[0]	Valid

Table 10-24: L1 data TLB format for Data Register 1

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

Table 10-25: L1 data TLB format for Data Register 2

Bit field	Description
[14]	Tagged MTE
[13]	FWB override
[12]	PBHA [3]
[11]	PBHA [2]
[10]	PBHA [1]
[9]	PBHA [0]
[8:0]	Physical Address [47:39]

10.2 L2 cache encodings

The L2 cache is 8-way set associative when configured for 2MB size and 12-way set associative when configured for 3MB size.

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 10-26: Neoverse™ V3 L2 cache tag location encoding for 2MB [2]

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

Table 10-27: Neoverse™ V3 L2 cache tag location encoding for 3MB [3]

Bit field of Xn	Description
[31:24]	RAMID = `0x10`
[23:22]	Reserved
[21:18]	Way(0-11)
[17:12]	Index[17:12]
[11:8]	XOR(Index[11:8], Way[3:0])
[7:6]	XOR(Index[7:6], Index[11:10], Way[3:2])
[5:0]	Reserved

Table 10-28: Neoverse™ V3 L2 cache data location encoding for 2MB [4]

Bit field of Xn	Description
[31:24]	RAMID = `0x11`
[23:21]	Reserved
[20:18]	Way (0-7)
[17:11]	XOR(Index[17:11], 7'b000100)
[10:8]	XOR(Index[10:8], Way[2:0])
[7]	XOR(Index[7], Index[11])
[6]	XOR(Index[6], Index[10], Way[2]
[5:4]	Physical address [5:4]
[3:0]	Reserved

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

Table 10-29: Neoverse™ V3 L2 cache data location encoding for 3MB [5]

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

Table 10-30: Neoverse™ V3 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 10-31: Neoverse™ V3 L2 victim location encoding for 2MB and 3 MB

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

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:

2MB or 3MB without coherent icache

Table 10-32: Data Register 0

Bit field	Description
[63:60]	ECC[3:0]
[59]	MPAM_PMG
[58:48]	MPAM_PARTID[10:0]

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

Bit field	Description
[47:46]	MPAM_SP[1:0]
[45:42]	PHBA[3:0]
[41:12]	Physical tag[47:18]
[11:10]	Security state[1:0] - {NSE, NS}
[9]	CopyAtHome
[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 10-33: Data Register 1

Bit field	Description
[63:4]	0
[3:0]	ECC[7:4]

Table 10-34: Data Register 2

Bit field	Description
[63:0]	0

2MB or 3MB with coherent icache

Table 10-35: Data register 0

Bit field	Description
[63:53]	MPAM_PARTID[10:0]
[52:51]	MPAM_SP[1:0]
[50:47]	Instruction cache valid
[46:43]	PHBA[3:0]

Bit field	Description
[42:13]	Physical tag[47:18]
[12:11]	Security state[1:0] - {NSE, NS}
[10]	CopyAtHome
[9:8]	Virtual Address[13:12]
[7]	Shareable
[6]	L1 data cache shared
[5]	L1 data cache valid
[4:3]	MTE state **0b00** Invalid **0b10** Clean **0b11** Dirty
[2:0]	L2 state **0b101** UniqueDirty **0b001** UniqueClean **0bx11** SharedClean **0bxx0** Invalid

Table 10-36: Data Register 1

Bit field	Description
[63:9]	0
[8:1]	ECC[7:0]
[0]	MPAM_PMG

Table 10-37: 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 data registers.

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

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

Table 10-39: L2 data RAM format for Data Register 1

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

Table 10-40: L2 data RAM format for Data Register 2

Bit field	Description
[63:20]	0
[19:12]	ECC for Data[127:64]
[11:4]	ECC for Data[63:0]
[3:0]	MTE tags

10.2.3 L2 TLB RAM returned data

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

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

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

Table 10-41: L2 TLB format for Instruction Register 0 if bit[6] is set to 0

Bit field	Description
[63:62]	Reserved
[61:20]	Physical address When bit[6] is 0: • [61:26] = PA[47:12] • [25:20] = Reserved

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 10-42: L2 TLB format for Instruction Register 0 if bit[6] is set to 1

Bit field	Description
[63:62]	Reserved
[61:20]	Physical address When bit[6] is 1: • [61:28] = PA[47:14] • [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 10-43: L2 TLB format for Instruction Register 1

Bit field	Description
[63]	ASID [0]
[62:59]	PBHA
[58]	Walk cache entry
[57:29]	Virtual address [48:20]
[28:24]	Reserved
[23:22]	Physical address space `0b00` : Secure `0b01` : Non-secure `0b10` : Root `0b11` : Realm
[21:11]	Reserved
[10]	nG, indicates a non-global page
[9]	Outer shareable
[8]	Inner shareable
[7:6]	Reserved
[5]	Outer allocate

Bit field	Description
[4:2]	Memory attributes: **0b000** Device nGnRnE **0b001** Device nGnRE **0b010** Device nGRE **0b011** Device GRE **0b100** Non-cacheable **0b101** Write-Back No-Allocate **0b110** Write-Back Transient **0b111** Write-Back Read-Allocate and Write-Allocate
[1:0]	Reserved

Table 10-44: L2 TLB format for Instruction Register 2

Bit field	Description
[63:34]	Reserved
[33:31]	MSID [2:0]: **0b000** Secure EL1 **0b001** Secure EL2 **0b010** Non-secure EL1 **0b011** Non-secure EL2 **0b101** EL3
[30:15]	VMID [15:0]
[14:0]	ASID [15:1]

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 data registers.

Table 10-45: Neoverse™ V3 L2 victim format for data register 0 for 2MB L2 cache

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

Table 10-46: Neoverse™ V3 L2 victim format for data register 1 for 2MB L2 cache

Bit field of Rd	Description
[63:0]	0

Table 10-47: Neoverse™ V3 L2 victim format for data register 2

Bit field of Rd	Description
[63:0]	0

Table 10-48: Neoverse™ V3 L2 victim format for data register 0 for 3MB L2 cache

Bit field of Rd	Description
[95:84]	Prefetch
[83:72]	Data source
[71:60]	Transient
[59:48]	Outer allocation hint
[47:24]	Pointer fill counter
[23:0]	Replacement[35:0]

Table 10-49: Neoverse™ V3 L2 victim format for data register 1

Bit field of Rd	Description
[63:0]	0

Table 10-50: Neoverse™ V3 L2 victim format for data register 2

Bit field of Rd	Description
[63:0]	0

11. RAS Extension support

The Neoverse™ V3 core supports the *Reliability, Availability, and Serviceability* (RAS) Extension, including all extensions up to Arm®v9.2-A.

In particular, the Neoverse™ V3 core supports these RAS Extension features:

- *Fault Handling Interrupts* (FHIs)
- *Error Recovery Interrupts* (ERIs)
- Poison attribute on bus transfers
- Cache protection with *Single Error Correct* (SEC) parity on the functional RAMs that only contain clean data. This includes the L1 instruction cache tag, L1 instruction cache data, and the *Memory Management Unit* (MMU) RAMs.
- Cache protection with *Single Error Correct*, *Double Error Detect* (SECDED), *Error Correcting Code* (ECC) on the functional RAMs that contain dirty data. This includes the L1 data cache tag, L1 data cache data, L2 cache tag, L2 cache data, and the L2 *Transaction Queue* (TQ) RAMs.
- Error Data Record registers to help software perform recovery actions
- Error injection capabilities to facilitate software and system debug
- The *Error Synchronization Barrier* (ESB) instruction to synchronize unrecoverable errors. When an `ESB` instruction is executed, the core ensures that all SError interrupts that are generated by instructions before the `ESB` are either taken or deferred. If the core cannot take the interrupt, it records the interrupt in the Deferred Interrupt Status Register DISR_EL1. See the *Arm® Architecture Reference Manual for A-profile architecture* for more information on DISR_EL1.
- Cache protection with Single Error Detect parity on MMU Granule Protection RAMs.
- Register File Parity protection with Single Error Detection on General and Vector Register Files.

Fault detection features are included in groups within the DSU-120 cluster and the Neoverse™ V3 core. Each group of fault detection features is referred to as a node. You can access each node by using either the System registers or the utility bus. The following nodes are implemented in the Neoverse™ V3 core and the DSU-120 cluster:

- Node 0 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® Architecture Reference Manual Supplement, Reliability, Availability, and Serviceability (RAS), for A-profile architecture*.

11.1 Cache protection behavior

The configuration of the *Reliability, Availability, and Serviceability* (RAS) Extension that is implemented in the Neoverse™ V3 core includes cache protection. In this case, the Neoverse™ V3 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Neoverse™ V3 core have the following capabilities:

SEC parity

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

SECDED ECC

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

The following table indicates which protection type is applied to each RAM in the Neoverse™ V3 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 memory	Parity or ECC support
L1 instruction cache data	SEC parity
L1 instruction cache tag	SEC parity
L1 data cache data	SECDED ECC
L1 data cache tag	SECDED ECC
Memory Management Unit (MMU) *Translation Cache* (TC)	SEC parity
Memory Management Unit (MMU) *Granule Protection Table* (GPT)	SEC Parity
L2 cache tag	SECDED ECC
L2 cache data	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, which might cause data corruption.

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

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

11.2 Register File Parity

The Neoverse™ V3 core can be configured at build time to include additional logic to check the integrity of flop-based storage entries in the *General Register File* (GRF) and *Vector Register File* (VRF) in the presence of potential *Single Event Upset* (SEU).

Including register file parity protection adds the logic to calculate and store appropriate parity bits for a register entry when the register is being written with new values. On the read path, the parity of the register entry being read is calculated and checked against stored parity bits; on a failed comparison, the error is signaled to be reported to RAS registers.

11.3 Error containment

The Neoverse™ V3 core supports error containment for data errors. This means that detected data errors are not silently propagated. Data errors are deferred using data poisoning, ensuring that a consumer is aware of the error. Uncorrectable L1 data cache tag errors and L2 cache tag errors are not containable.

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

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

Transient errors caused by *Single Event Upsets* (SEU) detected via register file parity are not containable.

11.4 Fault detection and reporting

When the Neoverse™ V3 core detects a fault, it raises a *Fault Handling Interrupt* (FHI) exception or an *Error Recovery Interrupt* (ERI) exception through the fault or the error signals. FHIs and ERIs are reflected in the *Reliability, Availability, and Serviceability* (RAS) registers, which are updated in the node that detects the errors.

Fault handling interrupts

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

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

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].

11.5 Error detection and reporting

When the Neoverse™ V3 core consumes an error, it raises different exceptions depending on the error type.

The Neoverse™ V3 core might raise:

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

11.5.1 Error reporting and performance monitoring

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

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

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

Related information

19.1 Performance monitors events on page 130

11.6 Error injection

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

Error injection uses the error detection and reporting registers to insert errors. The Neoverse™ V3 core can inject the following error types:

Corrected errors

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

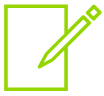
Deferred errors

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

Uncontainable errors

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

An error can be injected immediately or when a 32-bit counter reaches zero. You can control the value of the counter through the Error Pseudo-fault Generation Countdown Register, ERXPFGCDN_EL1 . The value of the counter decrements on a per clock cycle basis. See the *Arm® Architecture Reference Manual Supplement, Reliability, Availability, and Serviceability (RAS), for A-profile architecture* for more information about ERXPFGCDN_EL1.


Note

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

11.7 AArch64 RAS registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 11-2: RAS registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERXMISC1_EL1	3	0	C5	C5	1	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	0	C5	C5	2	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	0	C5	C5	3	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 3
DISR_EL1	3	0	C12	C1	1	See individual bit resets.	64-bit	Deferred Interrupt Status Register
VSESR_EL2	3	4	C5	C2	3	See individual bit resets.	64-bit	Virtual SError Exception Syndrome Register
VDISR_EL2	3	4	C12	C1	1	See individual bit resets.	64-bit	Virtual Deferred Interrupt Status Register
MFAR_EL3	3	6	C6	C0	5	See individual bit resets.	64-bit	Physical Fault Address Register (EL3)

11.8 External RAS registers summary

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

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

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

Table 11-3: RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	See individual bit resets.	64-bit	Error Record <n> Feature Register
0x8	ERR0CTLR	See individual bit resets.	64-bit	Error Record <n> Control Register
0x10	ERR0STATUS	See individual bit resets.	64-bit	Error Record <n> Primary Status Register
0x18	ERR0ADDR	See individual bit resets.	64-bit	Error Record <n> Address Register
0x20	ERR0MISC0	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 0
0x28	ERR0MISC1	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 1
0x30	ERR0MISC2	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 2
0x38	ERR0MISC3	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 3
0x800	ERR0PFGF	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Countdown Register

12. Utility bus

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

The utility bus provides access to the following system functions in the Neoverse™ V3 core:

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


Note

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

12.1 Base addresses for system components

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

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


Note

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

Table 12-1: Utility bus base addresses for system component registers

Base address, n is core instance number	Registers	Security state	Memory map
0x<n>9_0000	Core <n> AMU	Both	B.6 External AMU registers summary on page 1283
0x<n>A_0000	Core <n> RAS	Secure	A.12 AArch64 RAS registers summary on page 646
0x<n>B_0000	Core <n> MPMM	Secure	B.2 External PPM registers summary on page 1081

Base address, n is core instance number	Registers	Security state	Memory map
0x<n>D_0000 - 0x<n>F_0000	Reserved	-	-


Note

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

13. GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC Distributor connects to the Neoverse™ V3 core through a GIC CPU interface. The GIC CPU interface includes registers to mask, identify, and control the state of interrupts that are forwarded to the core.

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

The GICv4.1 architecture implemented in the Neoverse™ V3 core supports:

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

The GIC includes interrupt grouping functionality that supports:

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

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

13.1 Disable the GIC CPU interface

The Neoverse™ V3 core always includes the *Generic Interrupt Controller* (GIC) CPU interface. However, you can disable it to meet your requirements.

To disable the GIC CPU interface, assert the GICCDISABLE signal HIGH at reset. If you disable it this way, then you can use an external GIC IP to drive the interrupt signals (nFIQ, nIRQ). If the Neoverse™ V3 core is not integrated with an external GIC interrupt Distributor component (minimum GICv3 architecture) in the system, then you must disable the GIC CPU interface.

If you disable the GIC CPU interface, then:

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


Note

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

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

13.2 AArch64 GIC system registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 13-1: GIC system registers summary

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

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICV_IGRPEN0_EL1	3	0	C12	C12	6	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 0 Enable register
ICC_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 1 Enable register
ICV_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 1 Enable register
ICH_AP0R0_EL2	3	4	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R0_EL2	3	4	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 1 Registers
ICC_SRE_EL2	3	4	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable register (EL2)
ICH_HCR_EL2	3	4	C12	C11	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Control Register
ICH_VTR_EL2	3	4	C12	C11	1	See individual bit resets.	64-bit	Interrupt Controller VGIC Type Register
ICH_MISR_EL2	3	4	C12	C11	2	See individual bit resets.	64-bit	Interrupt Controller Maintenance Interrupt State Register
ICH_EISR_EL2	3	4	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller End of Interrupt Status Register
ICH_ELRSR_EL2	3	4	C12	C11	5	See individual bit resets.	64-bit	Interrupt Controller Empty List Register Status Register
ICH_VMCR_EL2	3	4	C12	C11	7	See individual bit resets.	64-bit	Interrupt Controller Virtual Machine Control Register
ICH_LR0_EL2	3	4	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR1_EL2	3	4	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR2_EL2	3	4	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_LR3_EL2	3	4	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICC_CTLR_EL3	3	6	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL3)
ICC_SRE_EL3	3	6	C12	C12	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable register (EL3)
ICC_IGRPEN1_EL3	3	6	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 1 Enable register (EL3)

14. Advanced SIMD and floating-point support

The Neoverse™ V3 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping.

The Neoverse™ V3 core floating-point implementation includes features up to Arm®v9.2-A. BFloat16 floating-point and Int8 matrix multiplication are part of these supported features.

The Neoverse™ V3 core implements all operations in hardware with support for all combinations of:

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

The Neoverse™ V3 core supports *Alternate Floating Point* behavior (FEAT_AFP), as part of Arm®v8.7-A and Arm®v9.2-A.

15. Scalable Vector Extensions support

The Neoverse™ V3 core supports the *Scalable Vector Extension* (SVE) and the *Scalable Vector Extension 2* (SVE2). SVE and SVE2 are intended to complement, not replace, AArch64 Advanced SIMD and floating-point functionality.

SVE is an optional extension introduced by the Armv8.2 architecture. The key features that SVE provides are:

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

The Neoverse™ V3 core implements a scalable vector length of 128 bits.

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

16. System control

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

The main functions of the system registers are:

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

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

16.1 AArch64 Generic System Control registers

The following summary table provides an overview of all Generic System Control registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 16-1: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ACTLR_EL1	3	0	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL1)
RGSR_EL1	3	0	C1	C0	5	See individual bit resets.	64-bit	Random Allocation Tag Seed Register.
GCR_EL1	3	0	C1	C0	6	See individual bit resets.	64-bit	Tag Control Register.
TTBR0_EL1	3	0	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL1)
TTBR1_EL1	3	0	C2	C0	1	See individual bit resets.	64-bit	Translation Table Base Register 1 (EL1)
TCR_EL1	3	0	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL1)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
APIAKeyLo_EL1	3	0	C2	C1	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Instruction (bits[63:0])
APIAKeyHi_EL1	3	0	C2	C1	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Instruction (bits[127:64])
APIBKeyLo_EL1	3	0	C2	C1	2	See individual bit resets.	64-bit	Pointer Authentication Key B for Instruction (bits[63:0])
APIBKeyHi_EL1	3	0	C2	C1	3	See individual bit resets.	64-bit	Pointer Authentication Key B for Instruction (bits[127:64])
APDAKeyLo_EL1	3	0	C2	C2	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Data (bits[63:0])
APDAKeyHi_EL1	3	0	C2	C2	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Data (bits[127:64])
APDBKeyLo_EL1	3	0	C2	C2	2	See individual bit resets.	64-bit	Pointer Authentication Key B for Data (bits[63:0])
APDBKeyHi_EL1	3	0	C2	C2	3	See individual bit resets.	64-bit	Pointer Authentication Key B for Data (bits[127:64])
APGAKeyLo_EL1	3	0	C2	C3	0	See individual bit resets.	64-bit	Pointer Authentication Key A for Code (bits[63:0])
APGAKeyHi_EL1	3	0	C2	C3	1	See individual bit resets.	64-bit	Pointer Authentication Key A for Code (bits[127:64])
SPSel	3	0	C4	C2	0	See individual bit resets.	64-bit	Stack Pointer Select
CurrentEL	3	0	C4	C2	2	See individual bit resets.	64-bit	Current Exception Level
PAN	3	0	C4	C2	3	See individual bit resets.	64-bit	Privileged Access Never
UAO	3	0	C4	C2	4	See individual bit resets.	64-bit	User Access Override
AFSR0_EL1	3	0	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	0	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL1)
ESR_EL1	3	0	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL1)
TFSR_EL1	3	0	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL1)
TFSRE0_EL1	3	0	C5	C6	1	See individual bit resets.	64-bit	Tag Fault Status Register (EL0).
FAR_EL1	3	0	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL1)
PAR_EL1	3	0	C7	C4	0	See individual bit resets.	64-bit	Physical Address Register
MAIR_EL1	3	0	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL1)
AMAIR_EL1	3	0	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
LORSA_EL1	3	0	C10	C4	0	See individual bit resets.	64-bit	LORegion Start Address (EL1)
LOREA_EL1	3	0	C10	C4	1	See individual bit resets.	64-bit	LORegion End Address (EL1)
LORN_EL1	3	0	C10	C4	2	See individual bit resets.	64-bit	LORegion Number (EL1)
LORC_EL1	3	0	C10	C4	3	See individual bit resets.	64-bit	LORegion Control (EL1)
LORID_EL1	3	0	C10	C4	7	See individual bit resets.	64-bit	LORegionID (EL1)
VBAR_EL1	3	0	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL1)
ISR_EL1	3	0	C12	C1	0	See individual bit resets.	64-bit	Interrupt Status Register
CONTEXTIDR_EL1	3	0	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL1)
TPIDR_EL1	3	0	C13	C0	4	See individual bit resets.	64-bit	EL1 Software Thread ID Register
SCXTNUM_EL1	3	0	C13	C0	7	See individual bit resets.	64-bit	EL1 Read/Write Software Context Number
IMP_CPUACTLR_EL1	3	0	C15	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL1)
IMP_CPUACTLR2_EL1	3	0	C15	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2 (EL1)
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 3 (EL1)
IMP_CPUACTLR4_EL1	3	0	C15	C1	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 4 (EL1)
IMP_CPUECTLR_EL1	3	0	C15	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	0	C15	C1	5	See individual bit resets.	64-bit	CPU Extended Control Register 2
IMP_CPUBUSQOS_EL1	3	0	C15	C1	7	See individual bit resets.	64-bit	CPU Bus QoS Register
IMP_CPUL2DIRTYLNCT_EL1	3	0	C15	C2	5	See individual bit resets.	64-bit	CPU L2 Dirty Line Count Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	See individual bit resets.	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	0	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL1)
IMP_CPUACTLR5_EL1	3	0	C15	C8	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 5 (EL1)
IMP_CPUACTLR6_EL1	3	0	C15	C8	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 6 (EL1)
IMP_CPUACTLR7_EL1	3	0	C15	C8	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 7 (EL1)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPUACTLR8_EL1	3	0	C15	C8	5	See individual bit resets.	64-bit	CPU Auxiliary Control Register 8 (EL1)
IMP_CPUACTLR9_EL1	3	0	C15	C8	6	See individual bit resets.	64-bit	CPU Auxiliary Control Register 9 (EL1)
AIDR_EL1	3	1	C0	C0	7	See individual bit resets.	64-bit	Auxiliary ID Register
RNDR	3	3	C2	C4	0	See individual bit resets.	64-bit	Random Number
RNDRRS	3	3	C2	C4	1	See individual bit resets.	64-bit	Reseeded Random Number
NZCV	3	3	C4	C2	0	See individual bit resets.	64-bit	Condition Flags
DAIF	3	3	C4	C2	1	See individual bit resets.	64-bit	Interrupt Mask Bits
DIT	3	3	C4	C2	5	See individual bit resets.	64-bit	Data Independent Timing
SSBS	3	3	C4	C2	6	See individual bit resets.	64-bit	Speculative Store Bypass Safe
TCO	3	3	C4	C2	7	See individual bit resets.	64-bit	Tag Check Override
FPCR	3	3	C4	C4	0	See individual bit resets.	64-bit	Floating-point Control Register
FPSR	3	3	C4	C4	1	See individual bit resets.	64-bit	Floating-point Status Register
TPIDR_EL0	3	3	C13	C0	2	See individual bit resets.	64-bit	EL0 Read/Write Software Thread ID Register
TPIDRRO_EL0	3	3	C13	C0	3	See individual bit resets.	64-bit	EL0 Read-Only Software Thread ID Register
SCXTNUM_EL0	3	3	C13	C0	7	See individual bit resets.	64-bit	EL0 Read/Write Software Context Number
ACTLR_EL2	3	4	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	4	C1	C1	7	See individual bit resets.	64-bit	Hypervisor Auxiliary Control Register
TTBR0_EL2	3	4	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL2)
TTBR1_EL2	3	4	C2	C0	1	See individual bit resets.	64-bit	Translation Table Base Register 1 (EL2)
TCR_EL2	3	4	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL2)
VTTBR_EL2	3	4	C2	C1	0	See individual bit resets.	64-bit	Virtualization Translation Table Base Register
VTCR_EL2	3	4	C2	C1	2	See individual bit resets.	64-bit	Virtualization Translation Control Register
VNCR_EL2	3	4	C2	C2	0	See individual bit resets.	64-bit	Virtual Nested Control Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
VSTTBR_EL2	3	4	C2	C6	0	See individual bit resets.	64-bit	Virtualization Secure Translation Table Base Register
VSTCR_EL2	3	4	C2	C6	2	See individual bit resets.	64-bit	Virtualization Secure Translation Control Register
AFSR0_EL2	3	4	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	4	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL2)
ESR_EL2	3	4	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL2)
TFSR_EL2	3	4	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL2)
FAR_EL2	3	4	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL2)
HPFAR_EL2	3	4	C6	C0	4	See individual bit resets.	64-bit	Hypervisor IPA Fault Address Register
MAIR_EL2	3	4	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL2)
AMAIR_EL2	3	4	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
VBAR_EL2	3	4	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL2)
CONTEXTIDR_EL2	3	4	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL2)
TPIDR_EL2	3	4	C13	C0	2	See individual bit resets.	64-bit	EL2 Software Thread ID Register
SCXTNUM_EL2	3	4	C13	C0	7	See individual bit resets.	64-bit	EL2 Read/Write Software Context Number
IMP_ATCR_EL2	3	4	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_AVTCR_EL2	3	4	C15	C7	1	See individual bit resets.	64-bit	CPU Virtualization Auxiliary Translation Control Register (EL2)
ACTLR_EL3	3	6	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL3)
SCR_EL3	3	6	C1	C1	0	See individual bit resets.	64-bit	Secure Configuration Register
CPTR_EL3	3	6	C1	C1	2	See individual bit resets.	64-bit	Architectural Feature Trap Register (EL3)
MDCR_EL3	3	6	C1	C3	1	See individual bit resets.	64-bit	Monitor Debug Configuration Register (EL3)
TTBR0_EL3	3	6	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL3)
TCR_EL3	3	6	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL3)
GPTBR_EL3	3	6	C2	C1	4	See individual bit resets.	64-bit	Granule Protection Table Base Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
GPCCR_EL3	3	6	C2	C1	6	See individual bit resets.	64-bit	Granule Protection Check Control Register (EL3)
AFSR0_EL3	3	6	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	6	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL3)
ESR_EL3	3	6	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL3)
TFSR_EL3	3	6	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL3)
FAR_EL3	3	6	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL3)
MAIR_EL3	3	6	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL3)
AMAIR_EL3	3	6	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
VBAR_EL3	3	6	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL3)
RVBAR_EL3	3	6	C12	C0	1	See individual bit resets.	64-bit	Reset Vector Base Address Register (if EL3 implemented)
RMR_EL3	3	6	C12	C0	2	See individual bit resets.	64-bit	Reset Management Register (EL3)
TPIDR_EL3	3	6	C13	C0	2	See individual bit resets.	64-bit	EL3 Software Thread ID Register
SCXTNUM_EL3	3	6	C13	C0	7	See individual bit resets.	64-bit	EL3 Read/Write Software Context Number
IMP_CPUL2SDIRTYLNCT_EL3	3	6	C15	C2	3	See individual bit resets.	64-bit	CPU L2 Secure Dirty Line Count Register
IMP_CPUACTLR_EL3	3	6	C15	C4	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL3)
IMP_ATCR_EL3	3	6	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL3)
IMP_CPUPSELR_EL3	3	6	C15	C8	0	See individual bit resets.	64-bit	Selected Instruction Private Select Register
IMP_CPUPCR_EL3	3	6	C15	C8	1	See individual bit resets.	64-bit	Selected Instruction Private Control Register
IMP_CPUPOR_EL3	3	6	C15	C8	2	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register
IMP_CPUPMR_EL3	3	6	C15	C8	3	See individual bit resets.	64-bit	Selected Instruction Private Mask Register
IMP_CPUPOR2_EL3	3	6	C15	C8	4	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register 2
IMP_CPUPMR2_EL3	3	6	C15	C8	5	See individual bit resets.	64-bit	Selected Instruction Private Mask Register 2
IMP_CPUPFR_EL3	3	6	C15	C8	6	See individual bit resets.	64-bit	Selected Instruction Private Flag Register

17. Random number generator support

The Neoverse™ V3 core can be configured to support two random number instructions introduced in the Arm®v8.5-A extension.

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

- MRS Xn, RNDR
- MRS Xn, RNDRRS

The Neoverse™ V3 core expects the *True Random Number Generator* (TRNG) and the *Deterministic Random Bit Generator* (DRBG) to be available as a memory-mapped peripheral and must be capable of the following requirements.

- Design the TRNG and DRBG as architecturally stipulated.
- Provide as many copies of TRNG and DRBG as is necessary to meet the overall bandwidth and latency requirements of the system.
- Reseed the DRBG from the TRNG when a RNDRRS instruction is received, as defined by the address encoding described in the Neoverse™ V3 core microarchitecture.
- Provide *Quality of Service* (QOS) managed access to DRBG bandwidth as architecturally defined.
- Provide access to each TRNG and DRBG block through a memory-mapped Dev-nGnRnE read (LDP Xreg).
- The address used by the LDP Xreg for RNDR and RNDRRS instructions is a physical-address defined as follows:
  - A combination of base register of 64K page (CPURNDBR_EL3[47:16]), PE-specific identifier (CPURNDPEID_EL3[10:0]), instruction-type.
  - CPURNDBR_EL3 [53:52] indicates the physical address space of the external RNG block accesses. The possible values are:
    - 0b00 Secure
    - 0b01 Non-secure
    - 0b10 Root
    - 0b11 Realm
  - RNDR physical address: {CPURNDBR_EL3[47:16], CPURNDPEID_EL3[10:0], 1'b0, 4'b0}
  - RNDRRS address: {CPURNDBR_EL3[47:16], CPURNDPEID_EL3[10:0], 1'b1, 4'b0}
- Set CPURNDBR_EL3[47:16] in each Neoverse™ V3 core to match the peripheral base of the TRNG and DRBG block corresponding to the core. The association of the core to TRNG and DRBG block is defined by the system integrator.
- The TRNG and DRBG block must correctly decode the read-address, using [15:5] as the unique core identifier for QOS guarantees.
- The TRNG and DRBG block must correctly decode bit [4] of the read-address, 0 specifying an RNDR instruction and 1 specifying an RNDRRS instruction.

- Upon receiving a RNDR or RNDRRS request, the TRNG and DRBG block must return a 64-bit random number in the first 64 bits and 1 in the second 64 bits. In the event that the DRBG block is unable to provide a random number within a system integrator defined timeframe, it will return 0 in the first and second 64 bits.
- In the event of a bus error, a RNDR or RNDRRS request will fail and the core will set the PSTATE.Z flag and assert a SEI.


Note

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

17.1 AArch64 Random number control registers summary

The following summary table provides an overview of all AArch64 random number generator registers in the core.

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

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

Table 17-1: registers summary

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

18. Debug

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

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

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

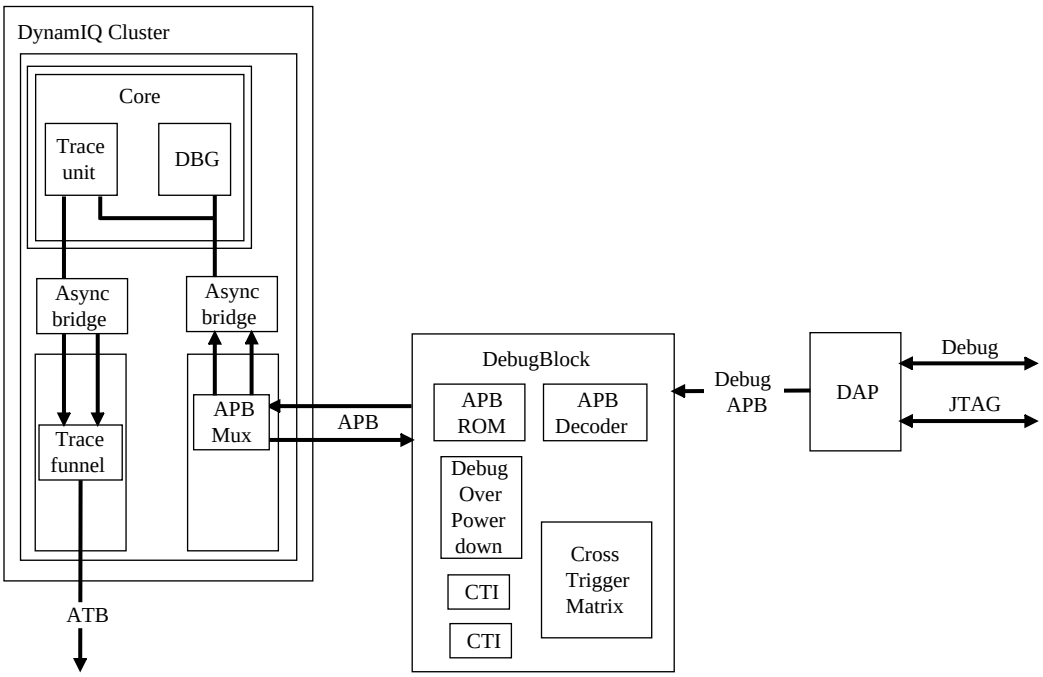
The connection between the cluster and the DebugBlock consists of a pair of *Advanced Peripheral Bus* (APB) interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register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

The debug system implements the following CoreSight debug components:

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

The following figure shows how the debug system is implemented with the DSU-120 cluster.

Figure 18-1: DSU-120 cluster debug components



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

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

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

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

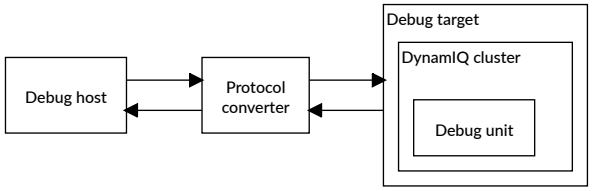
The Neoverse™ V3 core also supports direct access to internal memory, that is, cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use. See 10. Direct access to internal memory on page 78 for more information.

18.1 Supported debug methods

The DSU-120 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 18-2: External debug system



Debug host

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

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit. For DSU-120 based devices, the mechanism used to access the debug unit is based on the CoreSight architecture. The DSU-120 DebugBlock is accessed using an APB interface and the debug accesses are then directed to the Neoverse™ V3 core inside the DSU-120 cluster. An example of a debug target is a development system with a test chip or a silicon part with a Neoverse™ V3 core.

Debug unit

Helps debugging software that is running on the core:

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

With the debug unit, you can:

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

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

18.2 Debug register interfaces

The Neoverse™ V3 core implements the Arm®v9.2-A Debug architecture. It also supports the Arm®v8.4-A Debug architecture and Arm®v8.3-A Debug over powerdown.

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

Related information

5.7 Debug over powerdown on page 58

18.2.1 Core interfaces

System register access allows the Neoverse™ V3 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 DSU-120.

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-120.

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-120.

Statistical profiling

This function is system register based.

ELA registers

You can access the ELA registers using the APB slave port that connects into the DebugBlock of the DSU-120.

The ELA-600 is licensed separately.


Note

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

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

18.2.2 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.

18.2.3 Breakpoints and watchpoints

The Neoverse™ V3 core supports six breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC).

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

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

18.3 Debug events

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

The Neoverse™ V3 core responds to a debug event in one of the following ways:

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

In the Neoverse™ V3 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.

18.4 Debug memory map and debug signals

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

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

18.5 ROM table

The Neoverse™ V3 core includes a ROM table that contains a list of components in the system. Debuggers must use the ROM table to determine which CoreSight components are implemented.

The ROM table is a CoreSight debug related component that aids system debug along with CoreSight SoC and is for the Neoverse™ V3 core. The ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

The DSU-120 has its own ROM tables, one for the cluster and one for the DebugBlock, and has entry points in the cluster ROM table for the ROM tables belonging to each core. See *ROM tables* in the *Arm® DynamIQ™ Shared Unit-120 Technical Reference Manual* for more information.

The Neoverse™ V3 core ROM table includes the following entries:

Table 18-1: Core ROM table

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

Related information

18.10 External CoreROM registers on page 129

18.6 CoreSight component identification

Each component associated with the Neoverse™ V3 core has a unique set of CoreSight™ ID values. The following table shows these values.

Table 18-2: Neoverse™ V3 core CoreSight™ component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Core revision
DBG	0x04000BBD81	0xB105900D	0x15	0x47709A15	r0p0
PMU			0x16	0x47702A16	
Trace unit			0x13	0x47725A13	
ROM table			0x00	0x47700AF7	

18.7 CTI register identification values

The Neoverse™ V3 core *Cross Trigger Interface* (CTI) registers are located in the DebugBlock of the DSU-120.

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

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

Table 18-3: Core CTI register peripheral ID values

Register	Bitfield position	Bitfield name	Value
CTIPIDR1	[7:4]	DES_0	0b1011
	[3:0]	PART_1	0b1101
CTIPIDR0	[7:0]	PART_0	0x84

18.8 AArch64 Debug registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 18-4: Debug registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
OSDTRRX_EL1	2	0	C0	C0	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Receive
DBGBVR0_EL1	2	0	C0	C0	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR0_EL1	2	0	C0	C0	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR0_EL1	2	0	C0	C0	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR0_EL1	2	0	C0	C0	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR1_EL1	2	0	C0	C1	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR1_EL1	2	0	C0	C1	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR1_EL1	2	0	C0	C1	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR1_EL1	2	0	C0	C1	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
MDCCINT_EL1	2	0	C0	C2	0	See individual bit resets.	64-bit	Monitor DCC Interrupt Enable Register
MDSCR_EL1	2	0	C0	C2	2	See individual bit resets.	64-bit	Monitor Debug System Control Register
DBGBVR2_EL1	2	0	C0	C2	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR2_EL1	2	0	C0	C2	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR2_EL1	2	0	C0	C2	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR2_EL1	2	0	C0	C2	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
OSDTRTX_EL1	2	0	C0	C3	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Transmit
DBGBVR3_EL1	2	0	C0	C3	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR3_EL1	2	0	C0	C3	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR3_EL1	2	0	C0	C3	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR3_EL1	2	0	C0	C3	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR4_EL1	2	0	C0	C4	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR4_EL1	2	0	C0	C4	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGBVR5_EL1	2	0	C0	C5	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR5_EL1	2	0	C0	C5	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
OSECCR_EL1	2	0	C0	C6	2	See individual bit resets.	64-bit	OS Lock Exception Catch Control Register
MDRAR_EL1	2	0	C1	C0	0	See individual bit resets.	64-bit	Monitor Debug ROM Address Register
OSLAR_EL1	2	0	C1	C0	4	See individual bit resets.	64-bit	OS Lock Access Register
OSLSR_EL1	2	0	C1	C1	4	See individual bit resets.	64-bit	OS Lock Status Register
OSDLR_EL1	2	0	C1	C3	4	See individual bit resets.	64-bit	OS Double Lock Register
DBGPRCR_EL1	2	0	C1	C4	4	See individual bit resets.	64-bit	Debug Power Control Register
DBGCLAIMSET_EL1	2	0	C7	C8	6	See individual bit resets.	64-bit	Debug CLAIM Tag Set register
DBGCLAIMCLR_EL1	2	0	C7	C9	6	See individual bit resets.	64-bit	Debug CLAIM Tag Clear register
DBGAUTHSTATUS_EL1	2	0	C7	C14	6	See individual bit resets.	64-bit	Debug Authentication Status register
MDCCSR_EL0	2	3	C0	C1	0	See individual bit resets.	64-bit	Monitor DCC Status Register
DBGDTR_EL0	2	3	C0	C4	0	See individual bit resets.	64-bit	Debug Data Transfer Register, half-duplex
DBGDTRRX_EL0	2	3	C0	C5	0	See individual bit resets.	64-bit	Debug Data Transfer Register, Receive
DBGDTRTX_EL0	2	3	C0	C5	0	See individual bit resets.	64-bit	Debug Data Transfer Register, Transmit
TRFCR_EL1	3	0	C1	C2	1	See individual bit resets.	64-bit	Trace Filter Control Register (EL1)
MDCR_EL2	3	4	C1	C1	1	See individual bit resets.	64-bit	Monitor Debug Configuration Register (EL2)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRFCR_EL2	3	4	C1	C2	1	See individual bit resets.	64-bit	Trace Filter Control Register (EL2)
IMP_IDATA0_EL3	3	6	C15	C0	0	See individual bit resets.	64-bit	Instruction Register 0
IMP_IDATA1_EL3	3	6	C15	C0	1	See individual bit resets.	64-bit	Instruction Register 1
IMP_IDATA2_EL3	3	6	C15	C0	2	See individual bit resets.	64-bit	Instruction Register 2
IMP_DDATA0_EL3	3	6	C15	C1	0	See individual bit resets.	64-bit	Data Register 0
IMP_DDATA1_EL3	3	6	C15	C1	1	See individual bit resets.	64-bit	Data Register 1
IMP_DDATA2_EL3	3	6	C15	C1	2	See individual bit resets.	64-bit	Data Register 2

18.9 External Debug registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 18-5: Debug registers summary

Offset	Name	Reset	Width	Description
0x020	EDESR	See individual bit resets.	32-bit	External Debug Event Status Register
0x024	EDECR	See individual bit resets.	32-bit	External Debug Execution Control Register
0x030	EDWAR	See individual bit resets.	64-bit	External Debug Watchpoint Address Register
0x080	DBGDTRRX_EL0	See individual bit resets.	32-bit	Debug Data Transfer Register, Receive
0x084	EDITR	See individual bit resets.	32-bit	External Debug Instruction Transfer Register
0x088	EDSCR	See individual bit resets.	32-bit	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	See individual bit resets.	32-bit	Debug Data Transfer Register, Transmit
0x090	EDRCR	See individual bit resets.	32-bit	External Debug Reserve Control Register
0x098	EDECCR	See individual bit resets.	32-bit	External Debug Exception Catch Control Register
0x300	OSLAR_EL1	See individual bit resets.	32-bit	OS Lock Access Register
0x310	EDPRCR	See individual bit resets.	32-bit	External Debug Power/Reset Control Register
0x314	EDPRSR	See individual bit resets.	32-bit	External Debug Processor Status Register
0x400	DBGBVR0_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x408	DBGBCR0_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x410	DBGBVR1_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x418	DBGBCR1_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x420	DBGBVR2_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers

Offset	Name	Reset	Width	Description
0x428	DBGBCR2_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x430	DBGBVR3_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x438	DBGBCR3_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x440	DBGBVR4_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x448	DBGBCR4_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x450	DBGBVR5_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
0x458	DBGBCR5_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x800	DBGWVR0_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x808	DBGWCR0_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x810	DBGWVR1_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x818	DBGWCR1_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x820	DBGWVR2_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x828	DBGWCR2_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x830	DBGWVR3_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x838	DBGWCR3_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	See individual bit resets.	32-bit	Main ID Register
0xD20	EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
0xD28	EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
0xD48	EDDFR1 [31:0]	See individual bit resets.	32-bit	External Debug Feature Register 1
0xD4C	EDDFR1 [63:32]	See individual bit resets.	32-bit	External Debug Feature Register 1
0xD60	EDAA32PFR	See individual bit resets.	64-bit	External Debug Auxiliary Processor Feature Register
0xF00	EDITCTRL	See individual bit resets.	32-bit	External Debug Integration mode Control register
0xFA0	DBGCLAIMSET_EL1	See individual bit resets.	32-bit	Debug CLAIM Tag Set register
0xFA4	DBGCLAIMCLR_EL1	See individual bit resets.	32-bit	Debug CLAIM Tag Clear register
0xFA8	EDDEVAFF0	See individual bit resets.	32-bit	External Debug Device Affinity register 0
0xFAC	EDDEVAFF1	See individual bit resets.	32-bit	External Debug Device Affinity register 1
0xFB0	EDLAR	See individual bit resets.	32-bit	External Debug Lock Access Register
0xFB4	EDLSR	See individual bit resets.	32-bit	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	See individual bit resets.	32-bit	Debug Authentication Status register
0xFBC	EDDEVARCH	See individual bit resets.	32-bit	External Debug Device Architecture register
0xFC0	EDDEVID2	See individual bit resets.	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	See individual bit resets.	32-bit	External Debug Device ID register 1
0xFC8	EDDEVID	See individual bit resets.	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	See individual bit resets.	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	See individual bit resets.	32-bit	External Debug Component Identification Register 0

Offset	Name	Reset	Width	Description
0xFF4	EDCIDR1	See individual bit resets.	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	See individual bit resets.	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	See individual bit resets.	32-bit	External Debug Component Identification Register 3

18.10 External CoreROM registers

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

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

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

Table 18-6: CoreROM registers summary

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

19. Performance Monitors Extension support

The Neoverse™ V3 core implements the Performance Monitors Extension, including Arm®v8.4-A,Arm®v8.5-A and Arm®v8.7-A performance monitoring features.

The Neoverse™ V3 core *Performance Monitoring Unit* (PMU):

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

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

19.1 Performance monitors events

The Neoverse™ V3 core *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

Common Event PMU events

The following table shows the Neoverse™ V3 core performance monitors events that are generated and the numbers that the PMU uses to reference the events. The table also shows the bit position of each event on the event bus. Event numbers that are not listed are reserved.

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

See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for information about SVE-specific PMU events.


Note

Unless otherwise indicated, each of these events can be exported to the trace unit and selected in accordance with the *Arm® Embedded Trace Extension*.

Table 19-1: Common PMU events

Event number	Mnemonic	Description
0x0000	SW_INCR	**Instruction architecturally executed, Condition code check pass, software increment** Counts software writes to the PMSWINC_EL0 (software PMU increment) register. The PMSWINC_EL0 register is a manually updated counter for use by application software. This event could be used to measure any user program event, such as accesses to a particular data structure (by writing to the PMSWINC_EL0 register each time the data structure is accessed). To use the PMSWINC_EL0 register and event, developers must insert instructions that write to the PMSWINC_EL0 register into the source code. Since the SW_INCR event records writes to the PMSWINC_EL0 register, there is no need to do a read/increment/write sequence to the PMSWINC_EL0 register.
0x0001	L1I_CACHE_REFILL	**Level 1 instruction cache refill** Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch. Instruction fetches may include accessing multiple instructions, but the single cache line allocation is counted once.
0x0002	L1I_TLB_REFILL	**Level 1 instruction TLB refill** Counts level 1 instruction TLB refills from any Instruction fetch. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB.
0x0003	L1D_CACHE_REFILL	**Level 1 data cache refill** Counts level 1 data cache refills caused by speculatively executed load or store operations that missed in the level 1 data cache. This event only counts one event per cache line. This event does not count cache line allocations from preload instructions or from hardware cache prefetching. Since the caches are write-back only for this processor, there are no write-through cache accesses.
0x0004	L1D_CACHE	**Level 1 data cache access** Counts level 1 data cache accesses from any load/store operations. Atomic operations that resolve in the CPUs caches (near atomic operations) counts as both a write access and read access. Each access to a cache line is counted including the multiple accesses caused by single instructions such as LDM or STM. Each access to other level 1 data or unified memory structures, for example refill buffers, write buffers, and write-back buffers, are also counted. This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.

Event number	Mnemonic	Description
0x0005	L1D_TLB_REFILL	**Level 1 data TLB refill** Counts level 1 data TLB accesses that resulted in TLB refills. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count on an access from an AT(address translation) instruction. This event is the sum of the L1D_TLB_REFILL_RD and L1D_TLB_REFILL_WR events.
0x0008	INST_RETIRED	**Instruction architecturally executed** Counts instructions that have been architecturally executed.
0x0009	EXC_TAKEN	**Exception taken** Counts any taken architecturally visible exceptions such as IRQ, FIQ, SError, and other synchronous exceptions. Exceptions are counted whether or not they are taken locally.
0x000A	EXC_RETURN	**Instruction architecturally executed, Condition code check pass, exception return** Counts any architecturally executed exception return instructions. For example: AArch64: ERET
0x000B	CID_WRITE_RETIRED	**Instruction architecturally executed, Condition code check pass, write to CONTEXTIDR** Counts architecturally executed writes to the CONTEXTIDR_EL1 register, which usually contain the kernel PID and can be output with hardware trace.
0x000C	PC_WRITE_RETIRED	**Instruction architecturally executed, Condition code check pass, Software change of the PC** Counts branch instructions that caused a change of Program Counter, which effectively causes a change in the control flow of the program.
0x000D	BR_IMMED_RETIRED	**Branch instruction architecturally executed, immediate** Counts architecturally executed direct branches.
0x000E	BR_RETURN_RETIRED	**Branch instruction architecturally executed, procedure return, taken** Counts architecturally executed procedure returns.
0x0010	BR_MIS_PRED	**Branch instruction speculatively executed, mispredicted or not predicted** Counts branches which are speculatively executed and mispredicted.
0x0011	CPU_CYCLES	**Cycle** Counts CPU clock cycles (not timer cycles). The clock measured by this event is defined as the physical clock driving the CPU logic.
0x0012	BR_PRED	**Predictable branch instruction speculatively executed** Counts all speculatively executed branches.

Event number	Mnemonic	Description
0x0013	MEM_ACCESS	**Data memory access** Counts memory accesses issued by the CPU load store unit, where those accesses are issued due to load or store operations. This event counts memory accesses no matter whether the data is received from any level of cache hierarchy or external memory. If memory accesses are broken up into smaller transactions than what were specified in the load or store instructions, then the event counts those smaller memory transactions. Memory accesses generated by the following instructions or activity are not counted: Instruction fetches, Cache maintenance instructions, Translation table walks or prefetches, Memory prefetch operations. This event counts the sum of the MEM_ACCESS_RD and MEM_ACCESS_WR events.
0x0014	L1I_CACHE	**Level 1 instruction cache access** Counts instruction fetches which access the level 1 instruction cache. Instruction cache accesses caused by cache maintenance operations are not counted.
0x0015	L1D_CACHE_WB	**Level 1 data cache write-back** Counts write-backs of dirty data from the L1 data cache to the L2 cache. This occurs when either a dirty cache line is evicted from L1 data cache and allocated in the L2 cache or dirty data is written to the L2 and possibly to the next level of cache. This event counts both victim cache line evictions and cache write-backs from snoops or cache maintenance operations. The following cache operations are not counted: 1. Invalidations which do not result in data being transferred out of the L1 (such as evictions of clean data), 2. Full line writes which write to L2 without writing L1, such as write streaming mode. This event is the sum of the L1D_CACHE_WB_CLEAN and L1D_CACHE_WB_VICTIM events.
0x0016	L2D_CACHE	**Level 2 data cache access** Counts accesses to the level 2 cache due to data accesses. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the first level data cache or translation resolutions due to accesses. This event also counts write back of dirty data from level 1 data cache to the L2 cache. This CPU includes instruction cache accesses in this counter as L2I equivalent event was not implemented. This event is the sum of the L2D_CACHE_RD, L2D_CACHE_WR, L2D_CACHE_PRFM, L2D_CACHE_L1HWPRF, and L2D_CACHE_HWPRF events.
0x0017	L2D_CACHE_REFILL	**Level 2 data cache refill** Counts cache line refills into the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This CPU includes instruction cache refills in this counter as L2I equivalent event was not implemented. This event is the sum of L2D_CACHE_REFILL_RD, L2D_CACHE_REFILL_WR, L2D_CACHE_REFILL_L1HWPRF, L2D_CACHE_REFILL_HWPRF, and L2D_CACHE_REFILL_PRFM.

Event number	Mnemonic	Description
0x0018	L2D_CACHE_WB	**Level 2 data cache write-back** Counts write-backs of data from the L2 cache to outside the CPU. This includes snoops to the L2 (from other CPUs) which return data even if the snoops cause an invalidation. L2 cache line invalidations which do not write data outside the CPU and snoops which return data from an L1 cache are not counted. Data would not be written outside the cache when invalidating a clean cache line. This event is the sum of the L2D_CACHE_WB_VICTIM and L2D_CACHE_WB_CLEAN events.
0x0019	BUS_ACCESS	**Bus access** Counts memory transactions issued by the CPU to the external bus, including snoop requests and snoop responses. Each beat of data is counted individually.
0x001A	MEMORY_ERROR	**Local memory error** Counts any detected correctable or uncorrectable physical memory errors (ECC or parity) in protected CPUs RAMs. On the core, this event counts errors in the caches (including data and tag rams). Any detected memory error (from either a speculative and abandoned access, or an architecturally executed access) is counted. Note that errors are only detected when the actual protected memory is accessed by an operation.
0x001B	INST_SPEC	**Operation speculatively executed** Counts operations that have been speculatively executed.
0x001C	TTBR_WRITE_RETIRED	**Instruction architecturally executed, Condition code check pass, write to TTBR** Counts architectural writes to TTBR0/1_EL1. If virtualization host extensions are enabled (by setting the HCR_EL2.E2H bit to 1), then accesses to TTBR0/1_EL1 that are redirected to TTBR0/1_EL2, or accesses to TTBR0/1_EL12, are counted. TTBRn registers are typically updated when the kernel is swapping user-space threads or applications.
0x001D	BUS_CYCLES	**Bus cycle** Counts bus cycles in the CPU. Bus cycles represent a clock cycle in which a transaction could be sent or received on the interface from the CPU to the external bus. Since that interface is driven at the same clock speed as the CPU, this event is a duplicate of CPU_CYCLES.
0x001E	CHAIN	**Chain a pair of event counters** For odd-numbered counters, this event increments the count by one for each overflow of the preceding even-numbered counter. For even-numbered counters, there is no increment. This event is used when the even/odd pairs of registers are used as a single counter.
0x0021	BR_RETIRED	**Instruction architecturally executed, branch** Counts architecturally executed branches, whether the branch is taken or not. Instructions that explicitly write to the PC are also counted. Note that exception generating instructions, exception return instructions and context synchronization instructions are not counted.
0x0022	BR_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted** Counts branches counted by BR_RETIRED which were mispredicted and caused a pipeline flush.

Event number	Mnemonic	Description
0x0023	STALL_FRONTEND	**No operation sent for execution due to the frontend** Counts cycles when frontend could not send any micro-operations to the rename stage because of frontend resource stalls caused by fetch memory latency or branch prediction flow stalls. STALL_FRONTEND_SLOTS counts SLOTS during the cycle when this event counts. STALL_SLOT_FRONTEND will count SLOTS when this event is counted on this CPU.
0x0024	STALL_BACKEND	**No operation sent for execution due to the backend** Counts cycles whenever the rename unit is unable to send any micro-operations to the backend of the pipeline because of backend resource constraints. Backend resource constraints can include issue stage fullness, execution stage fullness, or other internal pipeline resource fullness. All the backend slots were empty during the cycle when this event counts.
0x0025	L1D_TLB	**Level 1 data TLB access** Counts level 1 data TLB accesses caused by any memory load or store operation. Note that load or store instructions can be broken up into multiple memory operations. This event does not count TLB maintenance operations.
0x0026	L1I_TLB	**Level 1 instruction TLB access** Counts level 1 instruction TLB accesses, whether the access hits or misses in the TLB. This event counts both demand accesses and prefetch or preload generated accesses. This event is a superset of the L1I_TLB_REFILL event.
0x002D	L2D_TLB_REFILL	**Level 2 data TLB refill** Counts level 2 TLB refills caused by memory operations from both data and instruction fetch, except for those caused by TLB maintenance operations and hardware prefetches. This event is the sum of the L2D_TLB_REFILL_RD and L2D_TLB_REFILL_WR events
0x002F	L2D_TLB	**Level 2 data TLB access** Counts level 2 TLB accesses except those caused by TLB maintenance operations. This event is the sum of the L2D_TLB_RD and L2D_TLB_WR events.
0x0031	REMOTE_ACCESS	**Access to another socket in a multi-socket system** Counts accesses to another chip, which is implemented as a different CMN mesh in the system. If the CHI bus response back to the core indicates that the data source is from another chip (mesh), then the counter is updated. If no data is returned, even if the system snoops another chip/mesh, then the counter is not updated.
0x0034	DTLB_WALK	**Data TLB access with at least one translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations. This event does not include prefetches.

Event number	Mnemonic	Description
0x0035	ITLB_WALK	**Instruction TLB access with at least one translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations. This event does not include prefetches.
0x0036	LL_CACHE_RD	**Last level cache access, read** Counts read transactions that were returned from outside the core cluster. This event counts for external last level cache when the system register CPUECTLR.EXTLLC bit is set, otherwise it counts for the L3 cache. This event counts read transactions returned from outside the core if those transactions are either hit in the system level cache or missed in the SLC and are returned from any other external sources. This event is a superset of the LL_CACHE_MISS_RD event.
0x0037	LL_CACHE_MISS_RD	**Last level cache miss, read** Counts read transactions that were returned from outside the core cluster but missed in the system level cache. This event counts for external last level cache when the system register CPUECTLR.EXTLLC bit is set, otherwise it counts for L3 cache. This event counts read transactions returned from outside the core if those transactions are missed in the System level Cache. The data source of the transaction is indicated by a field in the CHI transaction returning to the CPU. This event does not count reads caused by cache maintenance operations. This event is a subset of the LL_CACHE_RD event.
0x0039	L1D_CACHE_LMISS_RD	**Level 1 data cache long-latency read miss** Counts cache line refills into the level 1 data cache from any memory read operations, that incurred additional latency. Counts same as L1D_CACHE_REFILL_RD on this CPU.
0x003A	OP_RETIRED	**Micro-operation architecturally executed** Counts micro-operations that are architecturally executed. This is a count of number of micro-operations retired from the commit queue in a single cycle.
0x003B	OP_SPEC	**Micro-operation speculatively executed** Counts micro-operations speculatively executed. This is the count of the number of micro-operations dispatched in a cycle.
0x003C	STALL	**No operation sent for execution** Counts cycles when no operations are sent to the rename unit from the frontend or from the rename unit to the backend for any reason (either frontend or backend stall). This event is the sum of STALL_FRONTEND and STALL_BACKEND STALL is a sum of STALL_FRONTEND and STALL_BACKEND on this CPU

Event number	Mnemonic	Description
0x003D	STALL_SLOT_BACKEND	**No operation sent for execution on a Slot due to the backend** Counts slots per cycle in which no operations are sent from the rename unit to the backend due to backend resource constraints. STALL_BACKEND counts during the cycle when STALL_SLOT_BACKEND counts at least 1. STALL_BACKEND counts during the cycle when STALL_SLOT_BACKEND is SLOTS
0x003E	STALL_SLOT_FRONTEND	**No operation sent for execution on a Slot due to the frontend** Counts slots per cycle in which no operations are sent to the rename unit from the frontend due to frontend resource constraints. STALL_FRONTEND counts during the cycle when STALL_SLOT_FRONTEND is SLOTS
0x003F	STALL_SLOT	**No operation sent for execution on a Slot** Counts slots per cycle in which no operations are sent to the rename unit from the frontend or from the rename unit to the backend for any reason (either frontend or backend stall). STALL_SLOT is the sum of STALL_SLOT_FRONTEND and STALL_SLOT_BACKEND. STALL_SLOT is a sum of STALL_SLOT_FRONTEND and STALL_SLOT_BACKEND on this CPU
0x0040	L1D_CACHE_RD	**Level 1 data cache access, read** Counts level 1 data cache accesses from any load operation. Atomic load operations that resolve in the CPUs caches counts as both a write access and read access.
0x0041	L1D_CACHE_WR	**Level 1 data cache access, write** Counts level 1 data cache accesses generated by store operations. This event also counts accesses caused by a DC ZVA (data cache zero, specified by virtual address) instruction. Near atomic operations that resolve in the CPUs caches count as a write access and read access. This event is a subset of the L1D_CACHE event, except this event only counts memory-write operations.
0x0042	L1D_CACHE_REFILL_RD	**Level 1 data cache refill, read** Counts level 1 data cache refills caused by speculatively executed load instructions where the memory read operation misses in the level 1 data cache. This event only counts one event per cache line. This event is a subset of the L1D_CACHE_REFILL event, but only counts memory read operations. This event does not count reads caused by cache maintenance operations or preload instructions.
0x0043	L1D_CACHE_REFILL_WR	**Level 1 data cache refill, write** Counts level 1 data cache refills caused by speculatively executed store instructions where the memory write operation misses in the level 1 data cache. This event only counts one event per cache line. This event is a subset of the L1D_CACHE_REFILL event, but only counts memory write operations.
0x0044	L1D_CACHE_REFILL_INNER	**Level 1 data cache refill, inner** Counts level 1 data cache refills where the cache line data came from caches inside the immediate cluster of the core.

Event number	Mnemonic	Description
0x0045	L1D_CACHE_REFILL_OUTER	**Level 1 data cache refill, outer** Counts level 1 data cache refills for which the cache line data came from outside the immediate cluster of the core, like an SLC in the system interconnect or DRAM.
0x0046	L1D_CACHE_WB_VICTIM	**Level 1 data cache write-back, victim** Counts dirty cache line evictions from the level 1 data cache caused by a new cache line allocation. This event does not count evictions caused by cache maintenance operations. This event is a subset of the L1D_CACHE_WB event, but the event only counts write-backs that are a result of the line being allocated for an access made by the CPU.
0x0047	L1D_CACHE_WB_CLEAN	**Level 1 data cache write-back, cleaning and coherency** Counts write-backs from the level 1 data cache that are a result of a coherency operation made by another CPU. Event count includes cache maintenance operations. This event is a subset of the L1D_CACHE_WB event.
0x0048	L1D_CACHE_INVAL	**Level 1 data cache invalidate** Counts each explicit invalidation of a cache line in the level 1 data cache caused by: • Cache Maintenance Operations (CMO) that operate by a virtual address. • Broadcast cache coherency operations from another CPU in the system. This event does not count for the following conditions: 1. A cache refill invalidates a cache line. 2. A CMO which is executed on that CPU and invalidates a cache line specified by set/way. Note that CMOs that operate by set/way cannot be broadcast from one CPU to another.
0x004C	L1D_TLB_REFILL_RD	**Level 1 data TLB refill, read** Counts level 1 data TLB refills caused by memory read operations. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the translation table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count on an access from an Address Translation (AT) instruction. This event is a subset of the L1D_TLB_REFILL event.

Event number	Mnemonic	Description
0x004D	L1D_TLB_REFILL_WR	**Level 1 data TLB refill, write** Counts level 1 data TLB refills caused by data side memory write operations. If there are multiple misses in the TLB that are resolved by the refill, then this event only counts once. This event counts for refills caused by preload instructions or hardware prefetch accesses. This event counts regardless of whether the miss hits in L2 or results in a translation table walk. This event will not count if the table walk results in a fault (such as a translation or access fault), since there is no new translation created for the TLB. This event will not count with an access from an Address Translation (AT) instruction. This event is a subset of the L1D_TLB_REFILL event.
0x004E	L1D_TLB_RD	**Level 1 data TLB access, read** Counts level 1 data TLB accesses caused by memory read operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.
0x004F	L1D_TLB_WR	**Level 1 data TLB access, write** Counts any L1 data side TLB accesses caused by memory write operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.
0x0050	L2D_CACHE_RD	**Level 2 data cache access, read** Counts level 2 data cache accesses due to memory read operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This CPU includes instruction cache accesses in this counter as L2I equivalent event was not implemented. This event is a subset of the L2D_CACHE event, but this event only counts memory read operations.
0x0051	L2D_CACHE_WR	**Level 2 data cache access, write** Counts level 2 cache accesses due to memory write operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event is a subset of the L2D_CACHE event, but this event only counts memory write operations.
0x0052	L2D_CACHE_REFILL_RD	**Level 2 data cache refill, read** Counts refills for memory accesses due to memory read operation counted by L2D_CACHE_RD. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This CPU includes instruction cache refills in this counter as L2I equivalent event was not implemented. This event is a subset of the L2D_CACHE_REFILL event. This event does not count L2 refills caused by stashes into L2.
0x0053	L2D_CACHE_REFILL_WR	**Level 2 data cache refill, write** Counts refills for memory accesses due to memory write operation counted by L2D_CACHE_WR. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This event does not count on this CPU

Event number	Mnemonic	Description
0x0056	L2D_CACHE_WB_VICTIM	**Level 2 data cache write-back, victim** Counts evictions from the level 2 cache because of a line being allocated into the L2 cache. This event is a subset of the L2D_CACHE_WB event.
0x0057	L2D_CACHE_WB_CLEAN	**Level 2 data cache write-back, cleaning and coherency** Counts write-backs from the level 2 cache that are a result of either: 1. Cache maintenance operations, 2. Snoop responses or, 3. Direct cache transfers to another CPU due to a forwarding snoop request. This event is a subset of the L2D_CACHE_WB event.
0x0058	L2D_CACHE_INVAL	**Level 2 data cache invalidate** Counts each explicit invalidation of a cache line in the level 2 cache by cache maintenance operations that operate by a virtual address, or by external coherency operations. This event does not count if either: 1. A cache refill invalidates a cache line or, 2. A Cache Maintenance Operation (CMO), which invalidates a cache line specified by set/way, is executed on that CPU. CMOs that operate by set/way cannot be broadcast from one CPU to another.
0x005C	L2D_TLB_REFILL_RD	**Level 2 data TLB refill, read** Counts level 2 TLB refills caused by memory read operations from both data and instruction fetch except for those caused by TLB maintenance operations or hardware prefetches. This event is a subset of the L2D_TLB_REFILL event.
0x005D	L2D_TLB_REFILL_WR	**Level 2 data TLB refill, write** Counts level 2 TLB refills caused by memory write operations from both data and instruction fetch except for those caused by TLB maintenance operations. This event is a subset of the L2D_TLB_REFILL event.
0x005E	L2D_TLB_RD	**Level 2 data TLB access, read** Counts level 2 TLB accesses caused by memory read operations from both data and instruction fetch except for those caused by TLB maintenance operations. This event is a subset of the L2D_TLB event.
0x005F	L2D_TLB_WR	**Level 2 data TLB access, write** Counts level 2 TLB accesses caused by memory write operations from both data and instruction fetch except for those caused by TLB maintenance operations. This event is a subset of the L2D_TLB event.
0x0060	BUS_ACCESS_RD	**Bus access, read** Counts memory read transactions seen on the external bus. Each beat of data is counted individually.
0x0061	BUS_ACCESS_WR	**Bus access, write** Counts memory write transactions seen on the external bus. Each beat of data is counted individually.

Event number	Mnemonic	Description
0x0066	MEM_ACCESS_RD	**Data memory access, read** Counts memory accesses issued by the CPU due to load operations. The event counts any memory load access, no matter whether the data is received from any level of cache hierarchy or external memory. The event also counts atomic load operations. If memory accesses are broken up by the load/store unit into smaller transactions that are issued by the bus interface, then the event counts those smaller transactions. The following instructions are not counted: 1) Instruction fetches, 2) Cache maintenance instructions, 3) Translation table walks or prefetches, 4) Memory prefetch operations. This event is a subset of the MEM_ACCESS event but the event only counts memory-read operations.
0x0067	MEM_ACCESS_WR	**Data memory access, write** Counts memory accesses issued by the CPU due to store operations. The event counts any memory store access, no matter whether the data is located in any level of cache or external memory. The event also counts atomic load and store operations. If memory accesses are broken up by the load/store unit into smaller transactions that are issued by the bus interface, then the event counts those smaller transactions.
0x0068	UNALIGNED_LD_SPEC	**Unaligned access, read** Counts unaligned memory read operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses. The event does not count preload operations (PLD, PLI). This event is a subset of the UNALIGNED_LDST_SPEC event.
0x0069	UNALIGNED_ST_SPEC	**Unaligned access, write** Counts unaligned memory write operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses. This event is a subset of the UNALIGNED_LDST_SPEC event.
0x006A	UNALIGNED_LDST_SPEC	**Unaligned access** Counts unaligned memory operations issued by the CPU. This event counts unaligned accesses (as defined by the actual instruction), even if they are subsequently issued as multiple aligned accesses. This event is the sum of the UNALIGNED_ST_SPEC and UNALIGNED_LD_SPEC events.
0x006C	LDREX_SPEC	**Exclusive operation speculatively executed, Load-Exclusive** Counts Load-Exclusive operations that have been speculatively executed. For example: LDREX, LDX
0x006D	STREX_PASS_SPEC	**Exclusive operation speculatively executed, Store-Exclusive pass** Counts store-exclusive operations that have been speculatively executed and have successfully completed the store operation.
0x006E	STREX_FAIL_SPEC	**Exclusive operation speculatively executed, Store-Exclusive fail** Counts store-exclusive operations that have been speculatively executed and have not successfully completed the store operation.

Event number	Mnemonic	Description
0x006F	STREX_SPEC	**Exclusive operation speculatively executed, Store-Exclusive** Counts store-exclusive operations that have been speculatively executed. This event is the sum of STREX_PASS_SPEC and STREX_FAIL_SPEC events.
0x0070	LD_SPEC	**Operation speculatively executed, load** Counts speculatively executed load operations including Single Instruction Multiple Data (SIMD) load operations.
0x0071	ST_SPEC	**Operation speculatively executed, store** Counts speculatively executed store operations including Single Instruction Multiple Data (SIMD) store operations.
0x0072	LDST_SPEC	**Operation speculatively executed, load or store** Counts load and store operations that have been speculatively executed.
0x0073	DP_SPEC	**Operation speculatively executed, integer data processing** Counts speculatively executed logical or arithmetic instructions such as MOV/MVN operations.
0x0074	ASE_SPEC	**Operation speculatively executed, Advanced SIMD** Counts speculatively executed Advanced SIMD operations excluding load, store and move micro-operations that move data to or from SIMD (vector) registers.
0x0075	VFP_SPEC	**Operation speculatively executed, scalar floating-point** Counts speculatively executed floating point operations. This event does not count operations that move data to or from floating point (vector) registers.
0x0076	PC_WRITE_SPEC	**Operation speculatively executed, Software change of the PC** Counts speculatively executed operations which cause software changes of the PC. Those operations include all taken branch operations.
0x0077	CRYPTO_SPEC	**Operation speculatively executed, Cryptographic instruction** Counts speculatively executed cryptographic operations except for PMULL and VMULL operations.
0x0078	BR_IMMED_SPEC	**Branch speculatively executed, immediate branch** Counts direct branch operations which are speculatively executed.
0x0079	BR_RETURN_SPEC	**Branch speculatively executed, procedure return** Counts procedure return operations (RET, RETAA and RETAB) which are speculatively executed.
0x007A	BR_INDIRECT_SPEC	**Branch speculatively executed, indirect branch** Counts indirect branch operations including procedure returns, which are speculatively executed. This includes operations that force a software change of the PC, other than exception-generating operations and direct branch instructions. Some examples of the instructions counted by this event include BR Xn, RET, etc...
0x007C	ISB_SPEC	**Barrier speculatively executed, ISB** Counts ISB operations that are executed.
0x007D	DSB_SPEC	**Barrier speculatively executed, DSB** Counts DSB operations that are speculatively issued to Load/Store unit in the CPU.
0x007E	DMB_SPEC	**Barrier speculatively executed, DMB** Counts DMB operations that are speculatively issued to the Load/Store unit in the CPU. This event does not count implied barriers from load acquire/store release operations.

Event number	Mnemonic	Description
0x007F	CSDB_SPEC	**Barrier speculatively executed, CSDB** Counts CDSB operations that are speculatively issued to the Load/Store unit in the CPU. This event does not count implied barriers from load acquire/store release operations.
0x0081	EXC_UNDEF	**Exception taken, other synchronous** Counts the number of synchronous exceptions which are taken locally that are due to attempting to execute an instruction that is UNDEFINED. Attempting to execute instruction bit patterns that have not been allocated. Attempting to execute instructions when they are disabled. Attempting to execute instructions at an inappropriate Exception level. Attempting to execute an instruction when the value of PSTATE.IL is 1.
0x0082	EXC_SVC	**Exception taken, Supervisor Call** Counts SVC exceptions taken locally.
0x0083	EXC_PABORT	**Exception taken, Instruction Abort** Counts synchronous exceptions that are taken locally and caused by Instruction Aborts.
0x0084	EXC_DABORT	**Exception taken, Data Abort or SError** Counts exceptions that are taken locally and are caused by data aborts or SErrors. Conditions that could cause those exceptions are attempting to read or write memory where the MMU generates a fault, attempting to read or write memory with a misaligned address, interrupts from the nSEI inputs and internally generated SErrors.
0x0086	EXC_IRQ	**Exception taken, IRQ** Counts IRQ exceptions including the virtual IRQs that are taken locally.
0x0087	EXC_FIQ	**Exception taken, FIQ** Counts FIQ exceptions including the virtual FIQs that are taken locally.
0x0088	EXC_SMC	**Exception taken, Secure Monitor Call** Counts SMC exceptions take to EL3.
0x008A	EXC_HVC	**Exception taken, Hypervisor Call** Counts HVC exceptions taken to EL2.
0x008B	EXC_TRAP_PABORT	**Exception taken, Instruction Abort not Taken locally** Counts exceptions which are traps not taken locally and are caused by Instruction Aborts. For example, attempting to execute an instruction with a misaligned PC.
0x008C	EXC_TRAP_DABORT	**Exception taken, Data Abort or SError not Taken locally** Counts exceptions which are traps not taken locally and are caused by Data Aborts or SError interrupts. Conditions that could cause those exceptions are: 1. Attempting to read or write memory where the MMU generates a fault, 2. Attempting to read or write memory with a misaligned address, 3. Interrupts from the SEI input. 4. internally generated SErrors.
0x008D	EXC_TRAP_OTHER	**Exception taken, other traps not Taken locally** Counts the number of synchronous trap exceptions which are not taken locally and are not SVC, SMC, HVC, data aborts, Instruction Aborts, or interrupts.
0x008E	EXC_TRAP_IRQ	**Exception taken, IRQ not Taken locally** Counts IRQ exceptions including the virtual IRQs that are not taken locally.

Event number	Mnemonic	Description
0x008F	EXC_TRAP_FIQ	**Exception taken, FIQ not Taken locally** Counts FIQs which are not taken locally but taken from EL0, EL1, or EL2 to EL3 (which would be the normal behavior for FIQs when not executing in EL3).
0x0090	RC_LD_SPEC	**Release consistency operation speculatively executed, Load-Acquire** Counts any load acquire operations that are speculatively executed. For example: LDAR, LDARH, LDARB
0x0091	RC_ST_SPEC	**Release consistency operation speculatively executed, Store-Release** Counts any store release operations that are speculatively executed. For example: STLR, STLRH, STLRB
0x4000	SAMPLE_POP	**Statistical Profiling sample population** Counts statistical profiling sample population, the count of all operations that could be sampled but may or may not be chosen for sampling.
0x4001	SAMPLE_FEED	**Statistical Profiling sample taken** Counts statistical profiling samples taken for sampling.
0x4002	SAMPLE_FILTRATE	**Statistical Profiling sample taken and not removed by filtering** Counts statistical profiling samples taken which are not removed by filtering.
0x4003	SAMPLE_COLLISION	**Statistical Profiling sample collided with previous sample** Counts statistical profiling samples that have collided with a previous sample and so therefore not taken.
0x4004	CNT_CYCLES	**Constant frequency cycles** Increments at a constant frequency equal to the rate of increment of the System Counter, CNTPCT_EL0.
0x4005	STALL_BACKEND_MEM	**Memory stall cycles** Counts cycles when the backend is stalled because there is a pending demand load request in progress in the last level core cache. Last level cache in this CPU is Level 2, hence this event counts same as STALL_BACKEND_L2D
0x4006	L1I_CACHE_LMISS	**Level 1 instruction cache long-latency miss** Counts cache line refills into the level 1 instruction cache, that incurred additional latency. Counts the same as L1I_CACHE_REFILL in this CPU.
0x4009	L2D_CACHE_LMISS_RD	**Level 2 data cache long-latency read miss** Counts cache line refills into the level 2 unified cache from any memory read operations that incurred additional latency. Counts the same as L2D_CACHE_REFILL_RD in this CPU
0x4020	LDST_ALIGN_LAT	**Access with additional latency from alignment** Counts the number of memory read and write accesses in a cycle that incurred additional latency, due to the alignment of the address and the size of data being accessed, which results in store crossing a single cache line. This event is implemented as the sum of LD_ALIGN_LAT and ST_ALIGN_LAT on this CPU

Event number	Mnemonic	Description
0x4021	LD_ALIGN_LAT	**Load with additional latency from alignment** Counts the number of memory read accesses in a cycle that incurred additional latency, due to the alignment of the address and size of data being accessed, which results in load crossing a single cache line.
0x4022	ST_ALIGN_LAT	**Store with additional latency from alignment** Counts the number of memory write access in a cycle that incurred additional latency, due to the alignment of the address and size of data being accessed incurred additional latency.
0x4024	MEM_ACCESS_CHECKED	**Checked data memory access** Counts the number of memory read and write accesses counted by MEM_ACCESS that are tag checked by the Memory Tagging Extension (MTE). This event is implemented as the sum of MEM_ACCESS_CHECKED_RD and MEM_ACCESS_CHECKED_WR This event is implemented as the sum of MEM_ACCESS_CHECKED_RD and MEM_ACCESS_CHECKED_WR on this cpu
0x4025	MEM_ACCESS_CHECKED_RD	**Checked data memory access, read** Counts the number of memory read accesses in a cycle that are tag checked by the Memory Tagging Extension (MTE).
0x4026	MEM_ACCESS_CHECKED_WR	**Checked data memory access, write** Counts the number of memory write accesses in a cycle that is tag checked by the Memory Tagging Extension (MTE).
0x8004	SIMD_INST_SPEC	**Operation speculatively executed, SIMD** Counts speculatively executed operations that are SIMD or SVE vector operations or Advanced SIMD non-scalar operations.
0x8005	ASE_INST_SPEC	**Operation speculatively executed, Advanced SIMD** Counts speculatively executed Advanced SIMD operations.
0x8006	SVE_INST_SPEC	**Operation speculatively executed, SVE, including load and store** Counts speculatively executed operations that are SVE operations.
0x8014	FP_HP_SPEC	**Floating-point operation speculatively executed, half precision** Counts speculatively executed half precision floating point operations.
0x8018	FP_SP_SPEC	**Floating-point operation speculatively executed, single precision** Counts speculatively executed single precision floating point operations.
0x801C	FP_DP_SPEC	**Floating-point operation speculatively executed, double precision** Counts speculatively executed double precision floating point operations.
0x8040	INT_SPEC	**Integer operation speculatively executed** Counts speculatively executed integer arithmetic operations.
0x8074	SVE_PRED_SPEC	**Operation speculatively executed, SVE predicated** Counts speculatively executed predicated SVE operations.
0x8075	SVE_PRED_EMPTY_SPEC	**Operation speculatively executed, SVE predicated with no active predicates** Counts speculatively executed predicated SVE operations with no active predicate elements.
0x8076	SVE_PRED_FULL_SPEC	**Operation speculatively executed, SVE predicated with all active predicates** Counts speculatively executed predicated SVE operations with all predicate elements active.

Event number	Mnemonic	Description
0x8077	SVE_PRED_PARTIAL_SPEC	**Operation speculatively executed, SVE predicated with partially active predicates** Counts speculatively executed predicated SVE operations with at least one but not all active predicate elements.
0x8079	SVE_PRED_NOT_FULL_SPEC	**SVE predicated operations speculatively executed with no active or partially active predicates** Counts speculatively executed predicated SVE operations with at least one non active predicate elements.
0x8087	PRF_SPEC	**Operation speculatively executed, Prefetch** Counts speculatively executed operations that prefetch memory. For example: Scalar: PRFM, SVE: PRFB, PRFD, PRFH, or PRFW.
0x80BC	SVE_LDFF_SPEC	**Operation speculatively executed, SVE first-fault load** Counts speculatively executed SVE first fault or non-fault load operations.
0x80BD	SVE_LDFF_FAULT_SPEC	**Operation speculatively executed, SVE first-fault load which set FFR bit to 0b0** Counts speculatively executed SVE first fault or non-fault load operations that clear at least one bit in the FFR.
0x80C0	FP_SCALE_OPS_SPEC	**Scalable floating-point element ALU operations speculatively executed** Counts speculatively executed scalable single precision floating point operations.
0x80C1	FP_FIXED_OPS_SPEC	**Non-scalable floating-point element ALU operations speculatively executed** Counts speculatively executed non-scalable single precision floating point operations.
0x80E3	ASE_SVE_INT8_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 8-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type an 8-bit integer.
0x80E7	ASE_SVE_INT16_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 16-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 16-bit integer.
0x80EB	ASE_SVE_INT32_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 32-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 32-bit integer.
0x80EF	ASE_SVE_INT64_SPEC	**Integer operation speculatively executed, Advanced SIMD or SVE 64-bit** Counts speculatively executed Advanced SIMD or SVE integer operations with the largest data type a 64-bit integer.
0x8108	BR_IMMED_TAKEN_RETIRED	**Branch instruction architecturally executed, immediate, taken** Counts architecturally executed direct branches that were taken.
0x810C	BR_INDNR_TAKEN_RETIRED	**Branch instruction architecturally executed, indirect excluding procedure return, taken** Counts architecturally executed indirect branches excluding procedure returns that were taken.
0x8110	BR_IMMED_PRED_RETIRED	**Branch instruction architecturally executed, predicted immediate** Counts architecturally executed direct branches that were correctly predicted.
0x8111	BR_IMMED_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted immediate** Counts architecturally executed direct branches that were mispredicted and caused a pipeline flush.
0x8112	BR_IND_PRED_RETIRED	**Branch instruction architecturally executed, predicted indirect** Counts architecturally executed indirect branches including procedure returns that were correctly predicted.

Event number	Mnemonic	Description
0x8113	BR_IND_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted indirect** Counts architecturally executed indirect branches including procedure returns that were mispredicted and caused a pipeline flush.
0x8114	BR_RETURN_PRED_RETIRED	**Branch instruction architecturally executed, predicted procedure return** Counts architecturally executed procedure returns that were correctly predicted.
0x8115	BR_RETURN_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted procedure return** Counts architecturally executed procedure returns that were mispredicted and caused a pipeline flush.
0x8116	BR_INDNR_PRED_RETIRED	**Branch instruction architecturally executed, predicted indirect excluding procedure return** Counts architecturally executed indirect branches excluding procedure returns that were correctly predicted.
0x8117	BR_INDNR_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted indirect excluding procedure return** Counts architecturally executed indirect branches excluding procedure returns that were mispredicted and caused a pipeline flush.
0x8118	BR_TAKEN_PRED_RETIRED	**Branch instruction architecturally executed, predicted branch, taken** Counts architecturally executed branches that were taken and were correctly predicted.
0x8119	BR_TAKEN_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted branch, taken** Counts architecturally executed branches that were taken and were mispredicted causing a pipeline flush.
0x811A	BR_SKIP_PRED_RETIRED	**Branch instruction architecturally executed, predicted branch, not taken** Counts architecturally executed branches that were not taken and were correctly predicted.
0x811B	BR_SKIP_MIS_PRED_RETIRED	**Branch instruction architecturally executed, mispredicted branch, not taken** Counts architecturally executed branches that were not taken and were mispredicted causing a pipeline flush.
0x811C	BR_PRED_RETIRED	**Branch instruction architecturally executed, predicted branch** Counts branch instructions counted by BR_RETIRED which were correctly predicted.
0x811D	BR_IND_RETIRED	**Instruction architecturally executed, indirect branch** Counts architecturally executed indirect branches including procedure returns.
0x8120	INST_FETCH_PERCYC	**Event in progress, INST FETCH** Counts number of instruction fetches outstanding per cycle, which will provide an average latency of instruction fetch.
0x8121	MEM_ACCESS_RD_PERCYC	**Event in progress, MEM ACCESS RD** Counts the number of outstanding loads or memory read accesses per cycle.
0x8124	INST_FETCH	**Instruction memory access** Counts Instruction memory accesses that the PE makes.
0x8128	DTLB_WALK_PERCYC	**Event in progress, DTLB WALK** Counts the number of data translation table walks in progress per cycle.
0x8129	ITLB_WALK_PERCYC	**Event in progress, ITLB WALK** Counts the number of instruction translation table walks in progress per cycle.
0x812A	SAMPLE_FEED_BR	**Statisical Profiling sample taken, branch** Counts statistical profiling samples taken which are branches.

Event number	Mnemonic	Description
0x812B	SAMPLE_FEED_LD	**Statisical Profiling sample taken, load** Counts statistical profiling samples taken which are loads or load atomic operations.
0x812C	SAMPLE_FEED_ST	**Statisical Profiling sample taken, store** Counts statistical profiling samples taken which are stores or store atomic operations.
0x812D	SAMPLE_FEED_OP	**Statisical Profiling sample taken, matching operation type** Counts statistical profiling samples taken which are matching any operation type filters supported.
0x812E	SAMPLE_FEED_EVENT	**Statisical Profiling sample taken, matching events** Counts statistical profiling samples taken which are matching event packet filter constraints.
0x812F	SAMPLE_FEED_LAT	**Statisical Profiling sample taken, exceeding minimum latency** Counts statistical profiling samples taken which are exceeding minimum latency set by operation latency filter constraints.
0x8130	L1D_TLB_RW	**Level 1 data TLB demand access** Counts level 1 data TLB demand accesses caused by memory read or write operations. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.
0x8131	L1I_TLB_RD	**Level 1 instruction TLB demand access** Counts level 1 instruction TLB demand accesses whether the access hits or misses in the TLB.
0x8132	L1D_TLB_PRFM	**Level 1 data TLB software preload** Counts level 1 data TLB accesses generated by software prefetch or preload memory accesses. Load or store instructions can be broken into multiple memory operations. This event does not count TLB maintenance operations.
0x8133	L1I_TLB_PRFM	**Level 1 instruction TLB software preload** Counts level 1 instruction TLB accesses generated by software preload or prefetch instructions. This event counts whether the access hits or misses in the TLB. This event does not count TLB maintenance operations.
0x8134	DTLB_HWUPD	**Data TLB hardware update of translation table** Counts number of memory accesses triggered by a data translation table walk and performing an update of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that this event counts accesses triggered by software preloads, but not accesses triggered by hardware prefetchers.
0x8135	ITLB_HWUPD	**Instruction TLB hardware update of translation table** Counts number of memory accesses triggered by an instruction translation table walk and performing an update of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD.
0x8136	DTLB_STEP	**Data TLB translation table walk, step** Counts number of memory accesses triggered by a demand data translation table walk and performing a read of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that this event counts accesses triggered by software preloads, but not accesses triggered by hardware prefetchers.

Event number	Mnemonic	Description
0x8137	ITLB_STEP	**Instruction TLB translation table walk, step** Counts number of memory accesses triggered by an instruction translation table walk and performing a read of a translation table entry. Memory accesses are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD.
0x8138	DTLB_WALK_LARGE	**Data TLB large page translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and yielding a large page. The set of large pages is defined as all pages with a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. If DTLB_WALK_BLOCK is implemented, then it is an alias for this event in this family. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.
0x8139	ITLB_WALK_LARGE	**Instruction TLB large page translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a large page. The set of large pages is defined as all pages with a final size higher than or equal to 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_BLOCK event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813A	DTLB_WALK_SMALL	**Data TLB small page translation table walk** Counts number of data translation table walks caused by a miss in the L2 TLB and yielding a small page. The set of small pages is defined as all pages with a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. If DTLB_WALK_PAGE event is implemented, then it is an alias for this event in this family. Note that partial translations that cause a translation table walk are also counted. Also note that this event counts walks triggered by software preloads, but not walks triggered by hardware prefetchers, and that this event does not count walks triggered by TLB maintenance operations.
0x813B	ITLB_WALK_SMALL	**Instruction TLB small page translation table walk** Counts number of instruction translation table walks caused by a miss in the L2 TLB and yielding a small page. The set of small pages is defined as all pages with a final size lower than 2MB. Translation table walks that end up taking a translation fault are not counted, as the page size would be undefined in that case. In this family, this is equal to ITLB_WALK_PAGE event. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813C	DTLB_WALK_RW	**Data TLB demand access with at least one translation table walk** Counts number of demand data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.

Event number	Mnemonic	Description
0x813D	ITLB_WALK_RD	**Instruction TLB demand access with at least one translation table walk** Counts number of demand instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813E	DTLB_WALK_PRFM	**Data TLB software preload access with at least one translation table walk** Counts number of software prefetches or preloads generated data translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x813F	ITLB_WALK_PRFM	**Instruction TLB software preload access with at least one translation table walk** Counts number of software prefetches or preloads generated instruction translation table walks caused by a miss in the L2 TLB and performing at least one memory access. Translation table walks are counted even if the translation ended up taking a translation fault for reasons different than EPD, E0PD and NFD. Note that partial translations that cause a translation table walk are also counted. Also note that this event does not count walks triggered by TLB maintenance operations.
0x8140	L1D_CACHE_RW	**Level 1 data cache demand access** Counts level 1 data demand cache accesses from any load or store operation. Near atomic operations that resolve in the CPUs caches counts as both a write access and read access. This event is implemented as L1D_CACHE_RD + L1D_CACHE_WR
0x8141	L1I_CACHE_RD	**Level 1 instruction cache demand fetch** Counts demand instruction fetches which access the level 1 instruction cache.
0x8142	L1D_CACHE_PRFM	**Level 1 data cache software preload** Counts level 1 data cache accesses from software preload or prefetch instructions.
0x8143	L1I_CACHE_PRFM	**Level 1 instruction cache software preload** Counts instruction fetches generated by software preload or prefetch instructions which access the level 1 instruction cache.
0x8144	L1D_CACHE_MISS	**Level 1 data cache demand access miss** Counts cache line misses in the level 1 data cache.
0x8145	L1I_CACHE_HWPRF	**Level 1 instruction cache hardware prefetch** Counts instruction fetches which access the level 1 instruction cache generated by the hardware prefetcher.
0x8146	L1D_CACHE_REFILL_PRFM	**Level 1 data cache refill, software preload** Counts level 1 data cache refills where the cache line access was generated by software preload or prefetch instructions.
0x8147	L1I_CACHE_REFILL_PRFM	**Level 1 instruction cache refill, software preload** Counts cache line refills in the level 1 instruction cache caused by a missed instruction fetch generated by software preload or prefetch instructions. Instruction fetches may include accessing multiple instructions, but the single cache line allocation is counted once.

Event number	Mnemonic	Description
0x8148	L2D_CACHE_RW	**Level 2 data cache demand access** Counts level 2 cache demand accesses from any load/store operations. Level 2 cache is a unified cache for data and instruction accesses, accesses are for misses in the level 1 data cache or translation resolutions due to accesses. This CPU includes instruction cache accesses in this counter as L2I equivalent event was not implemented. This event is the sum of the L2D_CACHE_RD and L2D_CACHE_WR events.
0x814A	L2D_CACHE_PRFM	**Level 2 data cache software preload** Counts level 2 data cache accesses generated by software preload or prefetch instructions. This CPU includes instruction cache accesses in this counter as L2I equivalent event was not implemented.
0x814C	L2D_CACHE_MISS	**Level 2 data cache demand access miss** Counts cache line misses in the level 2 cache. Level 2 cache is a unified cache for data and instruction accesses. Accesses are for misses in the level 1 data cache or translation resolutions due to accesses. Counts same as L2D_CACHE_REFILL in this CPU
0x814E	L2D_CACHE_REFILL_PRFM	**Level 2 data cache refill, software preload** Counts refills due to accesses generated as a result of software preload or prefetch instructions as counted by L2D_CACHE_PRFM. This CPU includes instruction cache refills in this counter as L2I equivalent event was not implemented.
0x8154	L1D_CACHE_HWPRF	**Level 1 data cache hardware prefetch** Counts level 1 data cache accesses from any load/store operations generated by the hardware prefetcher.
0x8155	L2D_CACHE_HWPRF	**Level 2 data cache hardware prefetch** Counts level 2 data cache accesses generated by L2D hardware prefetchers.
0x8158	STALL_FRONTEND_MEMBOUND	**Frontend stall cycles, memory bound** Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the memory resources.
0x8159	STALL_FRONTEND_L1I	**Frontend stall cycles, level 1 instruction cache** Counts cycles when the frontend is stalled because there is an instruction fetch request pending in the level 1 instruction cache.
0x815B	STALL_FRONTEND_MEM	**Frontend stall cycles, last level PE cache or memory** Counts cycles when the frontend is stalled because there is an instruction fetch request pending in the last level core cache. Last level cache in this CPU is Level 2, hence this event counts rather than STALL_FRONTEND_L2I
0x815C	STALL_FRONTEND_TLB	**Frontend stall cycles, TLB** Counts when the frontend is stalled on any TLB misses being handled. This event also counts the TLB accesses made by hardware prefetches.

Event number	Mnemonic	Description
0x8160	STALL_FRONTEND_CPUBOUND	**Frontend stall cycles, processor bound** Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the CPU resources excluding memory resources.
0x8161	STALL_FRONTEND_FLOW	**Frontend stall cycles, flow control** Counts cycles when the frontend could not send any micro-operations to the rename stage due to resource constraints in the branch prediction unit.
0x8162	STALL_FRONTEND_FLUSH	**Frontend stall cycles, flush recovery** Counts cycles when the frontend could not send any micro-operations to the rename stage as the frontend is recovering from a machine flush or resteer. Example scenarios that cause a flush include branch mispredictions, taken exceptions, micro-architectural flush etc.
0x8164	STALL_BACKEND_MEMBOUND	**Backend stall cycles, memory bound** Counts cycles when the backend could not accept any micro-operations due to resource constraints in the memory resources.
0x8165	STALL_BACKEND_L1D	**Backend stall cycles, level 1 data cache** Counts cycles when the backend is stalled because there is a pending demand load request in progress in the level 1 data cache.
0x8166	STALL_BACKEND_L2D	**Backend stall cycles, level 2 data cache** Counts cycles when the backend is stalled because there is a pending demand load request in progress in the level 2 data cache.
0x8167	STALL_BACKEND_TLB	**Backend stall cycles, TLB** Counts cycles when the backend is stalled on any demand TLB misses being handled.
0x8168	STALL_BACKEND_ST	**Backend stall cycles, store** Counts cycles when the backend is stalled and there is a store that has not reached the pre-commit stage.
0x816A	STALL_BACKEND_CPUBOUND	**Backend stall cycles, processor bound** Counts cycles when the backend could not accept any micro-operations due to any resource constraints in the CPU excluding memory resources.
0x816B	STALL_BACKEND_BUSY	**Backend stall cycles, backend busy** Counts cycles when the backend could not accept any micro-operations because the issue queues are full to take any operations for execution.
0x816C	STALL_BACKEND_ILOCK	**Backend stall cycles, input dependency** Counts cycles when the backend could not accept any micro-operations due to resource constraints imposed by input dependency.
0x816D	STALL_BACKEND_RENAME	**Backend stall cycles, rename full** Counts cycles when backend is stalled even when operations are available from the frontend but at least one is not ready to be sent to the backend because no rename register is available.
0x81C0	L1I_CACHE_HIT_RD	**Level 1 instruction cache demand fetch hit** Counts demand instruction fetches that access the level 1 instruction cache and hit in the L1 instruction cache.
0x81D0	L1I_CACHE_HIT_RD_FPRFM	**Level 1 instruction cache demand fetch first hit, fetched by software preload** Counts demand instruction fetches that access the level 1 instruction cache that hit in the L1 instruction cache and the line was requested by a software prefetch.

Event number	Mnemonic	Description
0x81E0	L1I_CACHE_HIT_RD_FHWPRF	**Level 1 instruction cache demand fetch first hit, fetched by hardware prefetcher** Counts demand instruction fetches generated by hardware prefetch that access the level 1 instruction cache and hit in the L1 instruction cache.
0x8200	L1I_CACHE_HIT	**Level 1 instruction cache hit** Counts instruction fetches that access the level 1 instruction cache and hit in the level 1 instruction cache. Instruction cache accesses caused by cache maintenance operations are not counted.
0x8208	L1I_CACHE_HIT_PRFM	**Level 1 instruction cache software preload hit** Counts instruction fetches generated by software preload or prefetch instructions that access the level 1 instruction cache and hit in the level 1 instruction cache.
0x8240	L1I_LFB_HIT_RD	**Level 1 instruction cache demand fetch line-fill buffer hit** Counts demand instruction fetches that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache.
0x8250	L1I_LFB_HIT_RD_FPRFM	**Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by software preload** Counts demand instruction fetches generated by software prefetch instructions that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache.
0x8260	L1I_LFB_HIT_RD_FHWPRF	**Level 1 instruction cache demand fetch line-fill buffer first hit, recently fetched by hardware prefetcher** Counts demand instruction fetches generated by hardware prefetch that access the level 1 instruction cache and hit in a line that is in the process of being loaded into the level 1 instruction cache.

19.2 Performance monitors interrupts

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

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

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

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

19.3 External register access permissions

The Neoverse™ V3 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.

19.4 AArch64 Performance Monitors registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 19-2: Performance Monitors registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMCCNTR_EL0	3	3	C9	C13	0	See individual bit resets.	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	See individual bit resets.	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	See individual bit resets.	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	3	3	C9	C14	0	See individual bit resets.	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Set register
PMEVCNTR0_EL0	3	3	C14	C8	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	See individual bit resets.	64-bit	Performance Monitors Cycle Count Filter Register

19.5 External Performance Monitors registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 19-3: PMU registers summary

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

Offset	Name	Reset	Width	Description
0x204	PMPCSR [63:32]	See individual bit resets.	32-bit	Program Counter Sample Register
0x220	PMPCSR [31:0]	See individual bit resets.	32-bit	Program Counter Sample Register
0x224	PMPCSR [63:32]	See individual bit resets.	32-bit	Program Counter Sample Register
0x208	PMCID1SR	See individual bit resets.	32-bit	CONTEXTIDR_EL1 Sample Register
0x228	PMCID1SR	See individual bit resets.	32-bit	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	See individual bit resets.	32-bit	VMID Sample Register
0x22C	PMCID2SR	See individual bit resets.	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x418	PMEVTYPER6_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x41C	PMEVTYPER7_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x424	PMEVTYPER9_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x428	PMEVTYPER10_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x42C	PMEVTYPER11_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x434	PMEVTYPER13_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x438	PMEVTYPER14_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x43C	PMEVTYPER15_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x444	PMEVTYPER17_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x448	PMEVTYPER18_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x44C	PMEVTYPER19_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x450	PMEVTYPER20_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x454	PMEVTYPER21_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x458	PMEVTYPER22_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x45C	PMEVTYPER23_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x460	PMEVTYPER24_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x464	PMEVTYPER25_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x468	PMEVTYPER26_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x46C	PMEVTYPER27_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x470	PMEVTYPER28_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x474	PMEVTYPER29_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x478	PMEVTYPER30_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register

Offset	Name	Reset	Width	Description
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	See individual bit resets.	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x668	PMEVCNTSR9	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C0	PMEVCNTSR20	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C8	PMEVCNTSR21	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D0	PMEVCNTSR22	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D8	PMEVCNTSR23	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E0	PMEVCNTSR24	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E8	PMEVCNTSR25	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMEVCNTSR26	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F8	PMEVCNTSR27	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x700	PMEVCNTSR28	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x708	PMEVCNTSR29	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x710	PMEVCNTSR30	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0xC00	PMCNTENSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Set register
0xC20	PMCNTENCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Clear register
0xC40	PMINTENSET_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Set register
0xC60	PMINTENCLR_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Clear register
0xC80	PMOVSCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Clear register

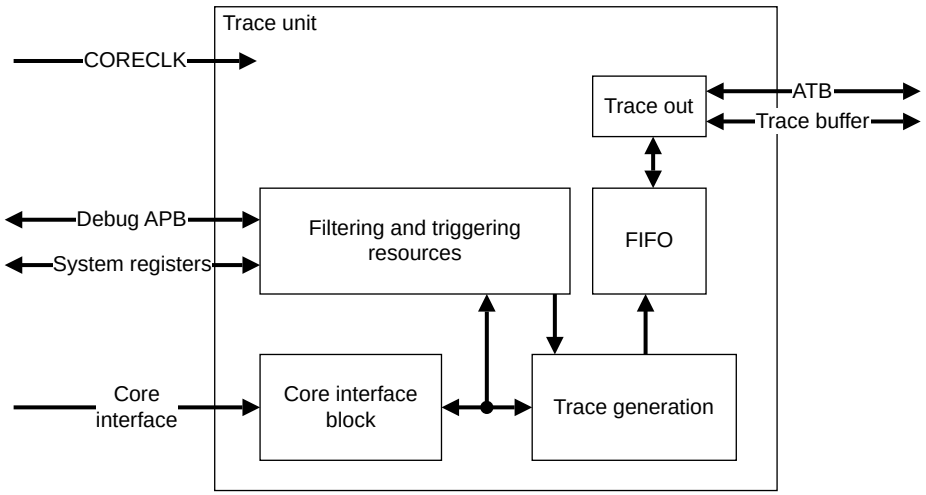
Offset	Name	Reset	Width	Description
0xCA0	PMSWINC_EL0	See individual bit resets.	32-bit	Performance Monitors Software Increment register
0xCC0	PMOVSSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Set register
0xE00	PMCFGR [31:0]	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	See individual bit resets.	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE30	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE40	PMMIR [31:0]	See individual bit resets.	32-bit	Performance Monitors Machine Identification Register
0xFA8	PMDEVAFF0	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 1
0xFB0	PMLAR	See individual bit resets.	32-bit	Performance Monitors Lock Access Register
0xFB4	PMLSR	See individual bit resets.	32-bit	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	See individual bit resets.	32-bit	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	See individual bit resets.	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	See individual bit resets.	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	See individual bit resets.	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 3

20. Embedded Trace Extension support

The Neoverse™ V3 core implements the *Embedded Trace Extension* (ETE). The trace unit performs real-time instruction flow tracing based on the ETE. The trace unit is a CoreSight component and is an integral part of the Arm real-time debug solution.

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

Figure 20-1: Trace unit components



Core interface

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

Trace generation

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

Filtering and triggering resources

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

FIFO

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

Trace out

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

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

20.1 Trace unit resources

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

The following table shows the trace unit resources, and indicates which of these resources the Neoverse™ V3 core implements.

Table 20-1: 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.

20.2 Trace unit generation options

The Neoverse™ V3 core trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the Neoverse™ V3 core trace unit.

Table 20-2: Trace unit generation options implemented

Description	Configuration
Instruction address size in bytes	8

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

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

20.3 Reset the trace unit

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

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

20.4 Program and read the trace unit registers

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

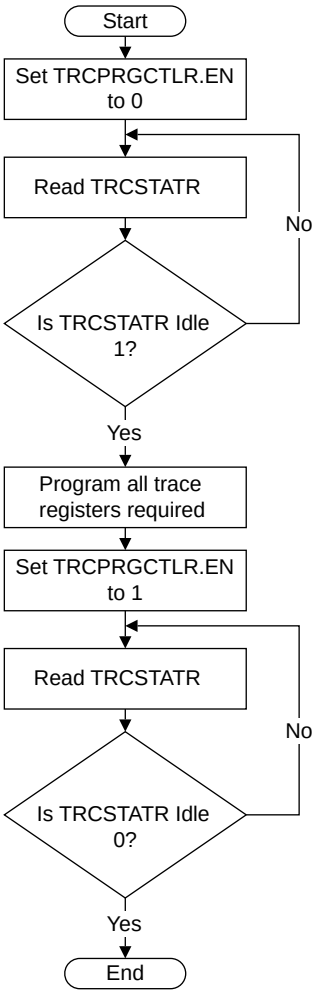
The core does not have to be in debug state when you program the trace unit registers. When you program the trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place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

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

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

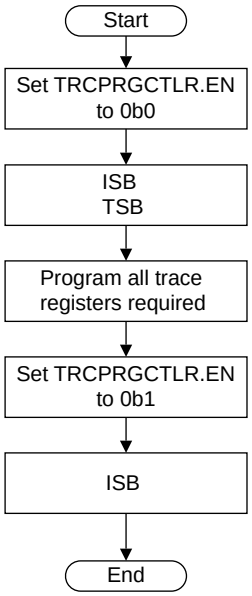
The following figure shows the flow for programming trace unit registers using the Debug APB interface:

Figure 20-2: 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 20-3: Programming trace registers using the System register interface



20.5 Trace unit register interfaces

The Neoverse™ V3 core supports an *Advanced Peripheral Bus* (APB) memory-mapped interface and a system register interface to trace unit registers.

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

20.6 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 Neoverse™ V3 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and trace unit function together.

Use of PMU events by the trace unit

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

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

Related information

19. Performance Monitors Extension support on page 130
19.1 Performance monitors events on page 130

20.7 Embedded Trace Extension events

The Neoverse™ V3 core trace unit collects events from other units in the design and uses numbers to reference these events.

Other than the events mentioned in 19.1 Performance monitors events on page 130, the events listed in the following table are also exported.

Table 20-3: ETE events

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

20.8 AArch64 Trace unit registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 20-4: Trace unit registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCTRACEIDR	2	1	C0	C0	1	See individual bit resets.	64-bit	Trace ID Register
TRCVICTLR	2	1	C0	C0	2	See individual bit resets.	64-bit	ViewInst Main Control Register
TRCSEQEVR0	2	1	C0	C0	4	See individual bit resets.	64-bit	Sequencer State Transition Control Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCCNTRLDVR0	2	1	C0	C0	5	See individual bit resets.	64-bit	Counter Reload Value Register <n>
TRCIDR8	2	1	C0	C0	6	See individual bit resets.	64-bit	ID Register 8
TRCIMSPEC0	2	1	C0	C0	7	See individual bit resets.	64-bit	IMP DEF Register 0
TRCPRGCTLR	2	1	C0	C1	0	See individual bit resets.	64-bit	Programming Control Register
TRCVIIECTLR	2	1	C0	C1	2	See individual bit resets.	64-bit	ViewInst Include/Exclude Control Register
TRCSEQEVR1	2	1	C0	C1	4	See individual bit resets.	64-bit	Sequencer State Transition Control Register <n>
TRCCNTRLDVR1	2	1	C0	C1	5	See individual bit resets.	64-bit	Counter Reload Value Register <n>
TRCIDR9	2	1	C0	C1	6	See individual bit resets.	64-bit	ID Register 9
TRCVISSCTLR	2	1	C0	C2	2	See individual bit resets.	64-bit	ViewInst Start/Stop Control Register
TRCSEQEVR2	2	1	C0	C2	4	See individual bit resets.	64-bit	Sequencer State Transition Control Register <n>
TRCIDR10	2	1	C0	C2	6	See individual bit resets.	64-bit	ID Register 10
TRCSTATR	2	1	C0	C3	0	See individual bit resets.	64-bit	Trace Status Register
TRCIDR11	2	1	C0	C3	6	See individual bit resets.	64-bit	ID Register 11
TRCCONFIGR	2	1	C0	C4	0	See individual bit resets.	64-bit	Trace Configuration Register
TRCCNTCTLR0	2	1	C0	C4	5	See individual bit resets.	64-bit	Counter Control Register <n>
TRCIDR12	2	1	C0	C4	6	See individual bit resets.	64-bit	ID Register 12
TRCCNTCTLR1	2	1	C0	C5	5	See individual bit resets.	64-bit	Counter Control Register <n>
TRCIDR13	2	1	C0	C5	6	See individual bit resets.	64-bit	ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	See individual bit resets.	64-bit	Auxiliary Control Register
TRCSEQRSTEVR	2	1	C0	C6	4	See individual bit resets.	64-bit	Sequencer Reset Control Register
TRCSEQSTR	2	1	C0	C7	4	See individual bit resets.	64-bit	Sequencer State Register
TRCEVENTCTL0R	2	1	C0	C8	0	See individual bit resets.	64-bit	Event Control 0 Register
TRCEXTINSELR0	2	1	C0	C8	4	See individual bit resets.	64-bit	External Input Select Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCCNTVR0	2	1	C0	C8	5	See individual bit resets.	64-bit	Counter Value Register <n>
TRCIDR0	2	1	C0	C8	7	See individual bit resets.	64-bit	ID Register 0
TRCEVENTCTL1R	2	1	C0	C9	0	See individual bit resets.	64-bit	Event Control 1 Register
TRCEXTINSELR1	2	1	C0	C9	4	See individual bit resets.	64-bit	External Input Select Register <n>
TRCCNTVR1	2	1	C0	C9	5	See individual bit resets.	64-bit	Counter Value Register <n>
TRCIDR1	2	1	C0	C9	7	See individual bit resets.	64-bit	ID Register 1
TRCRSR	2	1	C0	C10	0	See individual bit resets.	64-bit	Resources Status Register
TRCEXTINSELR2	2	1	C0	C10	4	See individual bit resets.	64-bit	External Input Select Register <n>
TRCIDR2	2	1	C0	C10	7	See individual bit resets.	64-bit	ID Register 2
TRCEXTINSELR3	2	1	C0	C11	4	See individual bit resets.	64-bit	External Input Select Register <n>
TRCIDR3	2	1	C0	C11	7	See individual bit resets.	64-bit	ID Register 3
TRCTSCTLR	2	1	C0	C12	0	See individual bit resets.	64-bit	Timestamp Control Register
TRCIDR4	2	1	C0	C12	7	See individual bit resets.	64-bit	ID Register 4
TRCSYNCPR	2	1	C0	C13	0	See individual bit resets.	64-bit	Synchronization Period Register
TRCIDR5	2	1	C0	C13	7	See individual bit resets.	64-bit	ID Register 5
TRCCCCTLR	2	1	C0	C14	0	See individual bit resets.	64-bit	Cycle Count Control Register
TRCIDR6	2	1	C0	C14	7	See individual bit resets.	64-bit	ID Register 6
TRCBBCTLR	2	1	C0	C15	0	See individual bit resets.	64-bit	Branch Broadcast Control Register
TRCIDR7	2	1	C0	C15	7	See individual bit resets.	64-bit	ID Register 7
TRCSSCCR0	2	1	C1	C0	2	See individual bit resets.	64-bit	Single-shot Comparator Control Register <n>
TRCOSLSR	2	1	C1	C1	4	See individual bit resets.	64-bit	Trace OS Lock Status Register
TRCRSCTLR2	2	1	C1	C2	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR3	2	1	C1	C3	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCRSCTLR4	2	1	C1	C4	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR5	2	1	C1	C5	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR6	2	1	C1	C6	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR7	2	1	C1	C7	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR8	2	1	C1	C8	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCSSCSR0	2	1	C1	C8	2	See individual bit resets.	64-bit	Single-shot Comparator Control Status Register <n>
TRCRSCTLR9	2	1	C1	C9	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR10	2	1	C1	C10	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR11	2	1	C1	C11	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR12	2	1	C1	C12	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR13	2	1	C1	C13	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR14	2	1	C1	C14	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR15	2	1	C1	C15	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCACVR0	2	1	C2	C0	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR0	2	1	C2	C0	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR1	2	1	C2	C2	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR1	2	1	C2	C2	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR2	2	1	C2	C4	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR2	2	1	C2	C4	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR3	2	1	C2	C6	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR3	2	1	C2	C6	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR4	2	1	C2	C8	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR4	2	1	C2	C8	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCACVR5	2	1	C2	C10	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR5	2	1	C2	C10	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR6	2	1	C2	C12	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR6	2	1	C2	C12	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR7	2	1	C2	C14	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR7	2	1	C2	C14	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCCIDCVR0	2	1	C3	C0	0	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>
TRCVMIDCVR0	2	1	C3	C0	1	See individual bit resets.	64-bit	Virtual Context Identifier Comparator Value Register <n>
TRCCIDCCTLR0	2	1	C3	C0	2	See individual bit resets.	64-bit	Context Identifier Comparator Control Register 0
TRCVMIDCCTLR0	2	1	C3	C2	2	See individual bit resets.	64-bit	Virtual Context Identifier Comparator Control Register 0
TRCDEVID	2	1	C7	C2	7	See individual bit resets.	64-bit	Device Configuration Register
TRCCLAIMSET	2	1	C7	C8	6	See individual bit resets.	64-bit	Claim Tag Set Register
TRCCLAIMCLR	2	1	C7	C9	6	See individual bit resets.	64-bit	Claim Tag Clear Register
TRCAUTHSTATUS	2	1	C7	C14	6	See individual bit resets.	64-bit	Authentication Status Register
TRCDEVARCH	2	1	C7	C15	6	See individual bit resets.	64-bit	Device Architecture Register

20.9 External Trace unit registers

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

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

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

Table 20-5: ETE registers summary

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	See individual bit resets.	32-bit	Auxiliary Control Register

Offset	Name	Reset	Width	Description
0x180	TRCIDR8	See individual bit resets.	32-bit	ID Register 8
0x184	TRCIDR9	See individual bit resets.	32-bit	ID Register 9
0x188	TRCIDR10	See individual bit resets.	32-bit	ID Register 10
0x18C	TRCIDR11	See individual bit resets.	32-bit	ID Register 11
0x190	TRCIDR12	See individual bit resets.	32-bit	ID Register 12
0x194	TRCIDR13	See individual bit resets.	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	See individual bit resets.	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	See individual bit resets.	32-bit	ID Register 0
0x1E4	TRCIDR1	See individual bit resets.	32-bit	ID Register 1
0x1E8	TRCIDR2	See individual bit resets.	32-bit	ID Register 2
0x1EC	TRCIDR3	See individual bit resets.	32-bit	ID Register 3
0x1F0	TRCIDR4	See individual bit resets.	32-bit	ID Register 4
0x1F4	TRCIDR5	See individual bit resets.	32-bit	ID Register 5
0x1F8	TRCIDR6	See individual bit resets.	32-bit	ID Register 6
0x1FC	TRCIDR7	See individual bit resets.	32-bit	ID Register 7
0xF00	TRCITCTRL	See individual bit resets.	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	See individual bit resets.	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	See individual bit resets.	32-bit	Claim Tag Clear Register
0xFBC	TRCDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	See individual bit resets.	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	See individual bit resets.	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	See individual bit resets.	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	See individual bit resets.	32-bit	Device Type Register
0xFD0	TRCPIDR4	See individual bit resets.	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	See individual bit resets.	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	See individual bit resets.	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	See individual bit resets.	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	See individual bit resets.	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	See individual bit resets.	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	See individual bit resets.	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	See individual bit resets.	32-bit	Component Identification Register 0
0xFF4	TRCCIDR1	See individual bit resets.	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	See individual bit resets.	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	See individual bit resets.	32-bit	Component Identification Register 3

21. Trace Buffer Extension support

The Neoverse™ V3 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.

21.1 Program and read the trace buffer registers

You can program and read the *TRace Buffer Extension* (TRBE) registers using the System register interface.

The core does not have to be in debug state when you program the TRBE registers. When you program the TRBE registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition. To disable the TRBE, use the TRBLIMITR_EL1.E bit.

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

21.2 Trace buffer register interface

The Neoverse™ V3 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.

22. Activity Monitors Extension support

The Neoverse™ V3 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitoring has features similar to performance monitoring features, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

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

The Neoverse™ V3 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.

22.1 Activity monitors access

The Neoverse™ V3 core supports access to activity monitors from the System register interface and supports read-only memory-mapped access using the utility bus interface.

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

Access enable bit

There are multiple access enable bits associated with the Activity Monitors System registers:

- AMUSERENR_EL0.EN controls access from EL0 to the Activity Monitors System registers
- CPTR_EL2.TAM controls access from EL0 and EL1 to the Activity Monitors System registers
- CPTR_EL3.TAM controls access from EL0, EL1, and EL2 to the Activity Monitors Extension System registers

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

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

System register access

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

External memory-mapped access

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

The base address for *Activity Monitoring Unit* (AMU) registers on the utility bus interface is `0x<n>90000`, where *n* is the Neoverse™ V3 core instance number in the DSU-120 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.

22.2 Activity monitors counters

The Neoverse™ V3 core implements four activity monitors counters, 0-3, and three auxiliary counters, 10-12 that map to specific *Activity Monitoring Unit* (AMU) events.

Each counter has the following characteristics:

- All events are counted in 64-bit wrapping counters that overflow when they wrap. There is no support for overflow status indication or interrupts.
- Any change in clock frequency, can affect any counter. For example, when a `WFI`, `WFE`, `WFIT`, or `WFET` instruction stops the clock.
- Events 0-3 and auxiliary events 10-12 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.

22.3 Activity monitors events

Activity monitors events in the Neoverse™ V3 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 22-1: 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

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR02	INSTR_RETIRED	0x0008	Instruction architecturally executed This counter increments for every instruction that is executed architecturally, including instructions that fail their condition code check.
AMEVCNTR03	STALL_BACKEND_MEM	0x4005	Memory stall cycles This counter counts cycles in which the core is unable to dispatch instructions from the front end to the back end due to a back end stall caused by a miss in the last level of cache within the core clock domain.
AMEVCNTR10	MPMM_THRESHOLD_GEAR0	0x0300	*Maximum Power Mitigation System* (MPMM) Gear 0 activity period threshold exceeded
AMEVCNTR11	MPMM_THRESHOLD_GEAR1	0x0301	*Maximum Power Mitigation System* (MPMM) Gear 1 activity period threshold exceeded
AMEVCNTR12	MPMM_THRESHOLD_GEAR2	0x0302	*Maximum Power Mitigation System* (MPMM) Gear 2 activity period threshold exceeded

Related information

5.5.1 Maximum Power Mitigation Mechanism on page 55

22.4 AArch64 Activity Monitors registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 22-2: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	See individual bit resets.	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	See individual bit resets.	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	See individual bit resets.	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	See individual bit resets.	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCNTENSET0_EL0	3	3	C13	C2	5	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 1
AMCNTENSET1_EL0	3	3	C13	C3	1	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR03_EL0	3	3	C13	C4	3	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1

22.5 External Activity Monitors registers

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 22-3: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x4	AMEVCNTR00 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x8	AMEVCNTR01 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0xC	AMEVCNTR01 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x14	AMEVCNTR02 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x1C	AMEVCNTR03 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x104	AMEVCNTR10 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x108	AMEVCNTR11 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x10C	AMEVCNTR11 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x114	AMEVCNTR12 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	See individual bit resets.	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	See individual bit resets.	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	See individual bit resets.	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	See individual bit resets.	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	See individual bit resets.	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	See individual bit resets.	32-bit	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	See individual bit resets.	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	See individual bit resets.	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 2

Offset	Name	Reset	Width	Description
0xFEC	AMPIDR3	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 3

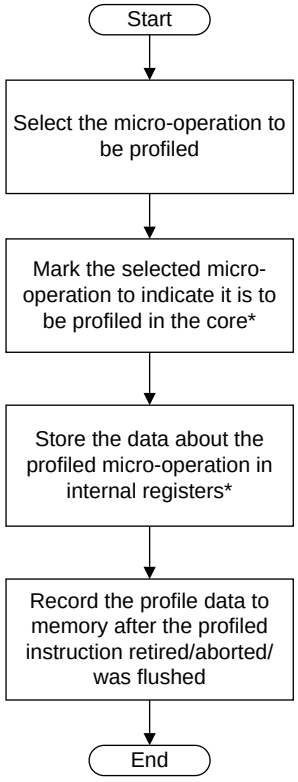
23. Statistical Profiling Extension support

The Neoverse™ V3 core implements the optional *Statistical Profiling Extension* (SPE) to the Arm®v8.7-A architecture. The SPE provides a statistical view of the performance characteristics of executed instructions that software writers can use to optimize their code for better performance.

The Neoverse™ V3 core profiles micro-operations to minimize the amount of logic necessary to support the SPE.

The following figure shows the SPE behavior in the Neoverse™ V3 core.

Figure 23-1: 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.

23.1 Statistical Profiling Extension events packet

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

The following table shows the events defined in the 32-bit events packet implemented in the Neoverse™ V3 core.

Table 23-1: 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

23.2 Statistical Profiling Extension data source packet

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

The following table shows the data source defined in the 8-bit data source packet implemented in the Neoverse™ V3 core.

Table 23-2: 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)

23.3 AArch64 Statistical Profiling Extension registers

The following summary table provides an overview of all Statistical Profiling Extension registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table 23-3: Statistical Profiling Extension registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSCR_EL1	3	0	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL1)
PMSNEVFR_EL1	3	0	C9	C9	1	See individual bit resets.	64-bit	Sampling Inverted Event Filter Register
PMSICR_EL1	3	0	C9	C9	2	See individual bit resets.	64-bit	Sampling Interval Counter Register
PMSIRR_EL1	3	0	C9	C9	3	See individual bit resets.	64-bit	Sampling Interval Reload Register
PMSFCR_EL1	3	0	C9	C9	4	See individual bit resets.	64-bit	Sampling Filter Control Register
PMSEVFR_EL1	3	0	C9	C9	5	See individual bit resets.	64-bit	Sampling Event Filter Register
PMSLATFR_EL1	3	0	C9	C9	6	See individual bit resets.	64-bit	Sampling Latency Filter Register
PMSIDR_EL1	3	0	C9	C9	7	See individual bit resets.	64-bit	Sampling Profiling ID Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMBLIMITR_EL1	3	0	C9	C10	0	See individual bit resets.	64-bit	Profiling Buffer Limit Address Register
PMBPTR_EL1	3	0	C9	C10	1	See individual bit resets.	64-bit	Profiling Buffer Write Pointer Register
PMBSR_EL1	3	0	C9	C10	3	See individual bit resets.	64-bit	Profiling Buffer Status/syndrome Register
PMBIDR_EL1	3	0	C9	C10	7	See individual bit resets.	64-bit	Profiling Buffer ID Register
PMSCR_EL2	3	4	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL2)

Appendix A AArch64 registers

This appendix contains the descriptions for the Neoverse™ V3 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 following summary table provides an overview of all Generic System Control registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-1: Generic System Control registers summary

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

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
VBAR_EL1	3	0	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL1)
ISR_EL1	3	0	C12	C1	0	See individual bit resets.	64-bit	Interrupt Status Register
CONTEXTIDR_EL1	3	0	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL1)
TPIDR_EL1	3	0	C13	C0	4	See individual bit resets.	64-bit	EL1 Software Thread ID Register
SCXTNUM_EL1	3	0	C13	C0	7	See individual bit resets.	64-bit	EL1 Read/Write Software Context Number
IMP_CPUACTLR_EL1	3	0	C15	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL1)
IMP_CPUACTLR2_EL1	3	0	C15	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2 (EL1)
IMP_CPUACTLR3_EL1	3	0	C15	C1	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 3 (EL1)
IMP_CPUACTLR4_EL1	3	0	C15	C1	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 4 (EL1)
IMP_CPUECTLR_EL1	3	0	C15	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	0	C15	C1	5	See individual bit resets.	64-bit	CPU Extended Control Register 2
IMP_CPUBUSQOS_EL1	3	0	C15	C1	7	See individual bit resets.	64-bit	CPU Bus QoS Register
IMP_CPUL2DIRTYLNCT_EL1	3	0	C15	C2	5	See individual bit resets.	64-bit	CPU L2 Dirty Line Count Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	See individual bit resets.	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	0	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL1)
IMP_CPUACTLR5_EL1	3	0	C15	C8	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 5 (EL1)
IMP_CPUACTLR6_EL1	3	0	C15	C8	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 6 (EL1)
IMP_CPUACTLR7_EL1	3	0	C15	C8	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 7 (EL1)
IMP_CPUACTLR8_EL1	3	0	C15	C8	5	See individual bit resets.	64-bit	CPU Auxiliary Control Register 8 (EL1)
IMP_CPUACTLR9_EL1	3	0	C15	C8	6	See individual bit resets.	64-bit	CPU Auxiliary Control Register 9 (EL1)
AIDR_EL1	3	1	C0	C0	7	See individual bit resets.	64-bit	Auxiliary ID Register
RNDR	3	3	C2	C4	0	See individual bit resets.	64-bit	Random Number
RNDRRS	3	3	C2	C4	1	See individual bit resets.	64-bit	Reseeded Random Number

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
NZCV	3	3	C4	C2	0	See individual bit resets.	64-bit	Condition Flags
DAIF	3	3	C4	C2	1	See individual bit resets.	64-bit	Interrupt Mask Bits
DIT	3	3	C4	C2	5	See individual bit resets.	64-bit	Data Independent Timing
SSBS	3	3	C4	C2	6	See individual bit resets.	64-bit	Speculative Store Bypass Safe
TCO	3	3	C4	C2	7	See individual bit resets.	64-bit	Tag Check Override
FPCR	3	3	C4	C4	0	See individual bit resets.	64-bit	Floating-point Control Register
FPSR	3	3	C4	C4	1	See individual bit resets.	64-bit	Floating-point Status Register
TPIDR_EL0	3	3	C13	C0	2	See individual bit resets.	64-bit	EL0 Read/Write Software Thread ID Register
TPIDRRO_EL0	3	3	C13	C0	3	See individual bit resets.	64-bit	EL0 Read-Only Software Thread ID Register
SCXTNUM_EL0	3	3	C13	C0	7	See individual bit resets.	64-bit	EL0 Read/Write Software Context Number
ACTLR_EL2	3	4	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	4	C1	C1	7	See individual bit resets.	64-bit	Hypervisor Auxiliary Control Register
TTBR0_EL2	3	4	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL2)
TTBR1_EL2	3	4	C2	C0	1	See individual bit resets.	64-bit	Translation Table Base Register 1 (EL2)
TCR_EL2	3	4	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL2)
VTTBR_EL2	3	4	C2	C1	0	See individual bit resets.	64-bit	Virtualization Translation Table Base Register
VTCR_EL2	3	4	C2	C1	2	See individual bit resets.	64-bit	Virtualization Translation Control Register
VNCR_EL2	3	4	C2	C2	0	See individual bit resets.	64-bit	Virtual Nested Control Register
VSTTBR_EL2	3	4	C2	C6	0	See individual bit resets.	64-bit	Virtualization Secure Translation Table Base Register
VSTCR_EL2	3	4	C2	C6	2	See individual bit resets.	64-bit	Virtualization Secure Translation Control Register
AFSR0_EL2	3	4	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	4	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL2)
ESR_EL2	3	4	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL2)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TFSR_EL2	3	4	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL2)
FAR_EL2	3	4	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL2)
HPFAR_EL2	3	4	C6	C0	4	See individual bit resets.	64-bit	Hypervisor IPA Fault Address Register
MAIR_EL2	3	4	C10	C2	0	See individual bit resets.	64-bit	Memory Attribute Indirection Register (EL2)
AMAIR_EL2	3	4	C10	C3	0	See individual bit resets.	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
VBAR_EL2	3	4	C12	C0	0	See individual bit resets.	64-bit	Vector Base Address Register (EL2)
CONTEXTIDR_EL2	3	4	C13	C0	1	See individual bit resets.	64-bit	Context ID Register (EL2)
TPIDR_EL2	3	4	C13	C0	2	See individual bit resets.	64-bit	EL2 Software Thread ID Register
SCXTNUM_EL2	3	4	C13	C0	7	See individual bit resets.	64-bit	EL2 Read/Write Software Context Number
IMP_ATCR_EL2	3	4	C15	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_AVTCR_EL2	3	4	C15	C7	1	See individual bit resets.	64-bit	CPU Virtualization Auxiliary Translation Control Register (EL2)
ACTLR_EL3	3	6	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL3)
SCR_EL3	3	6	C1	C1	0	See individual bit resets.	64-bit	Secure Configuration Register
CPTR_EL3	3	6	C1	C1	2	See individual bit resets.	64-bit	Architectural Feature Trap Register (EL3)
MDCR_EL3	3	6	C1	C3	1	See individual bit resets.	64-bit	Monitor Debug Configuration Register (EL3)
TTBR0_EL3	3	6	C2	C0	0	See individual bit resets.	64-bit	Translation Table Base Register 0 (EL3)
TCR_EL3	3	6	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL3)
GPTBR_EL3	3	6	C2	C1	4	See individual bit resets.	64-bit	Granule Protection Table Base Register
GPCCR_EL3	3	6	C2	C1	6	See individual bit resets.	64-bit	Granule Protection Check Control Register (EL3)
AFSR0_EL3	3	6	C5	C1	0	See individual bit resets.	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	6	C5	C1	1	See individual bit resets.	64-bit	Auxiliary Fault Status Register 1 (EL3)
ESR_EL3	3	6	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL3)
TFSR_EL3	3	6	C5	C6	0	See individual bit resets.	64-bit	Tag Fault Status Register (EL3)

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

A.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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-1: AArch64_actlr_el1 bit assignments

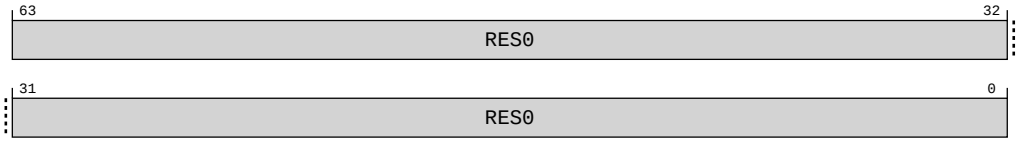


Table A-2: ACTLR_EL1 bit descriptions

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);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x118];
    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);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x118] = X[t, 64];
    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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-2: AArch64_afsr0_el1 bit assignments

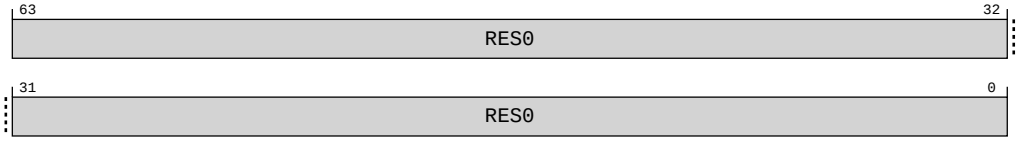


Table A-5: AFSR0_EL1 bit descriptions

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

Access

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

MRS <Xt>, AFSR0_EL1

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

MSR AFSR0_EL1, <Xt>

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

MRS <Xt>, AFSR0_EL12

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

MSR AFSR0_EL12, <Xt>

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

Accessibility

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

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x128];
    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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t, 64];
    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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x128];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
```

```
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x128] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR0_EL1 = X[t, 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-3: AArch64_afsr1_el1 bit assignments

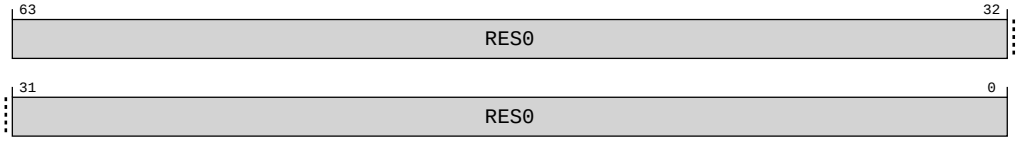


Table A-10: AFSR1_EL1 bit descriptions

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

Access

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

MRS <Xt>, AFSR1_EL1

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

MSR AFSR1_EL1, <Xt>

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

MRS <Xt>, AFSR1_EL12

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

MSR AFSR1_EL12, <Xt>

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

Accessibility

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

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x130];
    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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t, 64];
    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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x130];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x130] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSR1_EL1 = X[t, 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-4: AArch64_amair_el1 bit assignments

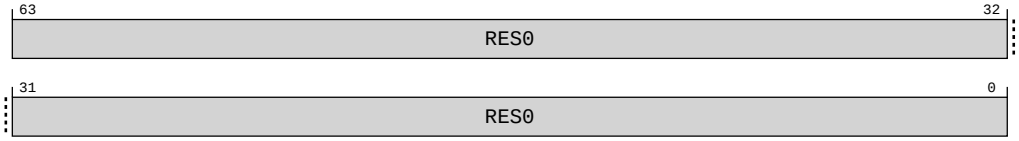


Table A-15: AMAIR_EL1 bit descriptions

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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x148];
    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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t, 64];
    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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x148];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        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
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x148] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t, 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-5: AArch64_lorid_el1 bit assignments

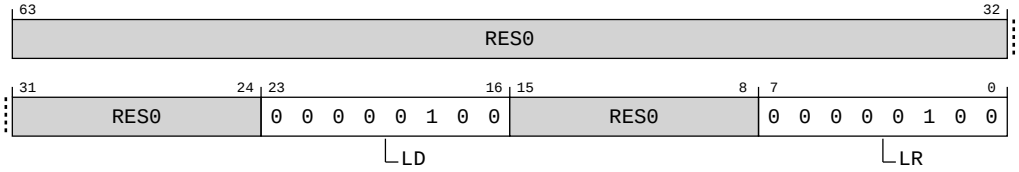


Table A-20: LORID_EL1 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.LORID_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TLOR == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-6: AArch64_imp_cpuactlr_el1 bit assignments

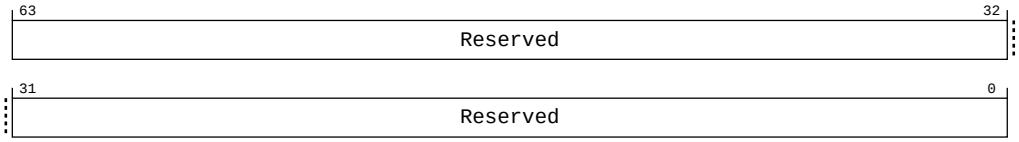


Table A-22: IMP_CPUACTLR_EL1 bit descriptions

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-7: AArch64_imp_cpuactlr2_el1 bit assignments

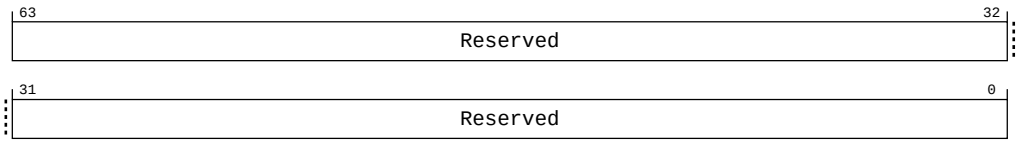


Table A-25: IMP_CPUACTLR2_EL1 bit descriptions

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-8: AArch64_imp_cpuactlr3_el1 bit assignments

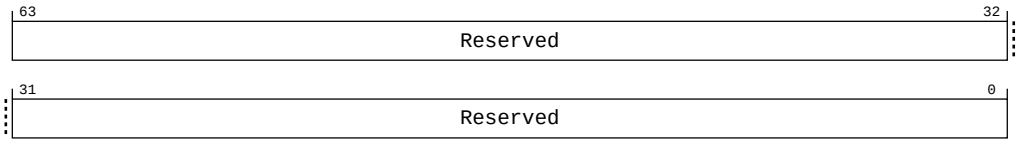


Table A-28: IMP_CPUACTLR3_EL1 bit descriptions

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 xxxx
```

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-9: AArch64_imp_cpuactlr4_el1 bit assignments

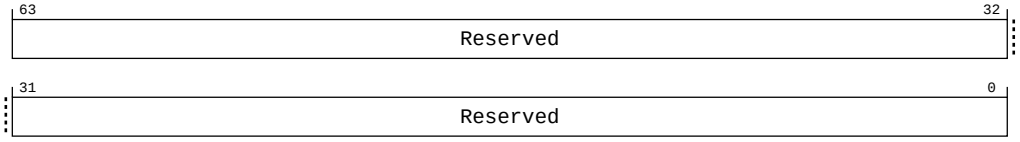


Table A-31: IMP_CPUACTLR4_EL1 bit descriptions

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

```
1100 0000 0000 0xxx 00xx 0100 0000 0x11 0100 0000 0101 01xx 0x11 xx00 xx00 0xx0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-10: AArch64_imp_cpuectlr_el1 bit assignments

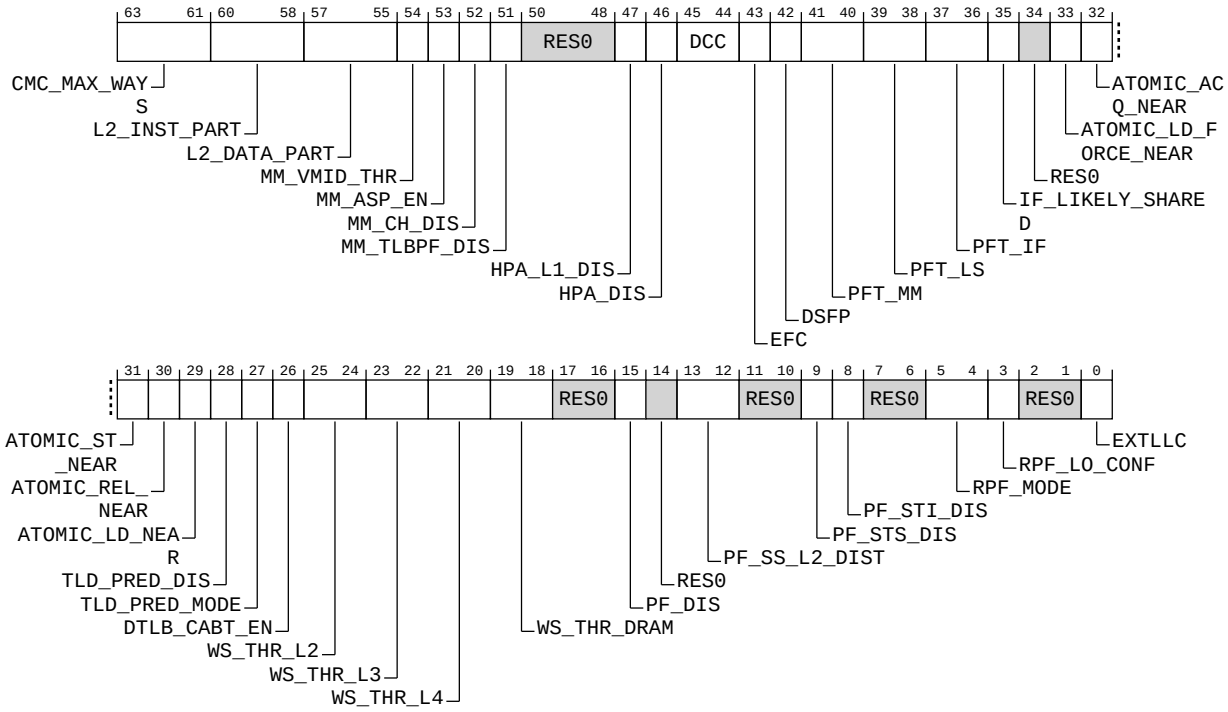


Table A-34: IMP_CPUECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:61]	CMC_MAX_WAYS	Maximum number of ways of the the L2 used by CMC. The possible values are: **0b000** CMC disabled **0b001** CMC can use 1 way **0b010** CMC can use 2 ways **0b011** CMC can use 3 ways **0b100** CMC can use 4 ways **0b101** CMC can use 5 ways **0b110** CMC can use 6 ways. This is the reset value **0b111** Reserved	0b110

Bits	Name	Description	Reset
[60:58]	L2_INST_PART	Partition the L2 cache for Instruction. The possible values are: **0b000** No ways reserved for instructions. This is the reset value **0b001** Reserve 1 way for instruction. Only instruction fetches can allocate way 7 (2MB) or 11 (3MB) **0b010** Reserve 2 ways for instruction. Only instruction fetches can allocate ways 7:6 (2MB) or 11:10 (3MB) **0b011** Reserve 3 ways for instruction. Only instruction fetches can allocate ways 7:5 (2MB) or 11:9 (3MB) **0b100** Reserve 4 ways for instruction. Only instruction fetches can allocate ways 7:4 (2MB) or 11:8 (3MB) **0b101** Reserve 5 ways for instruction. Only instruction fetches can allocate ways 7:3 (2MB) or 11:7 (3MB) **0b110** Reserve 6 ways for instruction. Only instruction fetches can allocate ways 7:2 (2MB) or 11:6 (3MB) **0b111** Reserved	0b000
[57:55]	L2_DATA_PART	Reserve L2 capacity for data accesses. The possible values are: **0b000** No ways reserved for data. This is the reset value **0b001** Reserve 1 way for data. Only data accesses can allocate way 0 **0b010** Reserve 2 ways for data. Only data accesses can allocate ways 1:0 **0b011** Reserve 3 ways for data. Only data accesses can allocate ways 2:0 **0b100** Reserve 4 ways for data. Only data accesses can allocate ways 3:0 **0b101** Reserve 5 ways for data. Only data accesses can allocate ways 4:0 **0b110** Reserve 6 ways for data. Only data accesses can allocate ways 5:0 **0b111** Reserved	0b000

Bits	Name	Description	Reset
[54]	MM_VMID_THR	VMID filter threshold. The possible values are: **0b0** VMID filter flush after 16 unique VMID allocations to the MMU Translation Cache. This is the default value. **0b1** VMID filter flush after 32 unique VMID allocations to the MMU Translation Cache	0b0
[53]	MM_ASP_EN	Disables allocation of splintered pages in L2 TLB. The possible values are: **0b0** Enables allocation of splintered pages in the L2 TLB. This is the default value. **0b1** Disables allocation of splintered pages in the L2 TLB.	0b0
[52]	MM_CH_DIS	Disables use of contiguous hint. The possible values are: **0b0** Enables use of contiguous hint. This is the default value. **0b1** Disables use of contiguous hint.	0b0
[51]	MM_TLBPF_DIS	Disables TLB prefetcher. The possible values are: **0b0** Enables TLB prefetcher. This is the default value. **0b1** Disables TLB prefetcher.	0b0
[50:48]	**RES0**	Reserved	**RES0**
[47]	HPA_L1_DIS	Disables hardware page aggregation in L1 TLBs. The possible values are: **0b0** Enables hardware page aggregation in L1 TLBs. This is the default value. **0b1** Disables hardware page aggregation in L1 TLBs.	0b0
[46]	HPA_DIS	Disable Hardware page aggregation. The possible values are: **0b0** Enables hardware page aggregation. This is the default value. **0b1** Disables hardware page aggregation.	0b0

Bits	Name	Description	Reset
[45:44]	DCC	Controls whether evictions of clean cache-lines send data on the CHI interface. Set this based on whether there is a cache on the path to memory. The possible values are: **0b00** Disables sending data when clean cache-lines are evicted. **0b01** Enables sending WriteEvictFull transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. **0b10** Enables sending WriteEvictOrEvict transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. **0b11** Enables sending WriteEvictOrEvict transactions when Unique Clean or Shared Clean cache-lines are evicted.	xx
[43]	EFC	Eviction Flush Control (EFC). Controls whether hardware cache flushes and DC CISW instruction send data when evicting clan and dirty cache line. If it is known that data is likely to be used soon by another core, setting this bit can improve system performance. The possible values are: **0b0** Disables cache allocation in downstream caches when hardware cache flushes or DC CISW instructions evict a clean or cache line. Downstream Snoop Filter Present (DFSP) controls the sending of Evict transactions **0b1** Enables cache allocation in downstream caches when hardware cache flushes or DC CISW instructions evict a clean or dirty cache line. Downstream Cache Control (DCC) controls the sending of data. DSFP controls the sending of Evict transactions.	0b0
[42]	DSFP	Downstream Snoop Filter Present (DSFP). Enables sending Evict transactions on the CHI interface when clean lines are evicted without data. You must enable this if there is at least one snoop filter in the path to memory **0b0** Disables sending Evict transactions when clean cachelines are evicted without data **0b1** Enables sending of Evict transaction when clean cachelines are evicted without data.	0b1

Bits	Name	Description	Reset
[41:40]	PFT_MM	DRAM prefetch using PrefetchTgt transactions for tablewalk requests. The possible values are: **0b00** Disable prefetchtgt generation for requests from the Memory Management unit (MMU). This is the default value. **0b01** Conservatively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable. **0b10** Agressively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable. **0b11** Always generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable.	0b00
[39:38]	PFT_LS	DRAM prefetch using PrefetchTgt transactions for load and store requests. The possible values are: **0b00** Disable prefetchtgt generation for requests from the Load-Store unit (LS). This is the default value. **0b01** Conservatively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable. **0b10** Agressively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable. **0b11** Always generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable.	0b00
[37:36]	PFT_IF	DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. The possible values are: **0b00** Disable prefetchtgt generation for requests from the Instruction Fetch unit (IF). This is the default value. **0b01** Conservatively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable. **0b10** Agressively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable. **0b11** Always generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable.	0b00

Bits	Name	Description	Reset
[35]	IF_LIKELY_SHARED	Instruction fetch Shared state control. The possible values are: **0b0** Instruction fetch requests do not assert TXREQ LikelyShared. This is the reset value. **0b1** Instruction fetch requests assert TXREQ LikelyShared and request a SharedClean copy of data.	0b0
[34]	**RES0**	Reserved	**RES0**
[33]	ATOMIC_LD_FORCE_NEAR	A load atomic (including SWP & CAS) instruction to WB memory will be performed near. The possible values are: **0b0** Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **0b1** Load-atomic will be performed near by bringing the line into the L1D Cache. This is the default value.	0b1
[32]	ATOMIC_ACQ_NEAR	An atomic instruction to WB memory with acquire semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Acquire-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **0b1** Acquire-atomic will make up to 1 fill request to perform near. This is the default value.	0b1
[31]	ATOMIC_ST_NEAR	A store atomic instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Store-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value. **0b1** Store-atomic will make up to 1 fill request to perform near.	0b0
[30]	ATOMIC_REL_NEAR	An atomic instruction to WB memory with release semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Release-atomic is near if cache line is already Exclusive, otherwise make far atomic request. **0b1** Release-atomic will make up to 1 fill request to perform near. This is the default value.	0b1
[29]	ATOMIC_LD_NEAR	A load atomic (including SWP & CAS) instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are: **0b0** Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value. **0b1** Load-atomic will make up to 1 fill request to perform near.	0b0

Bits	Name	Description	Reset
[28]	TLD_PRED_DIS	Disable Transient Load Prediction. The possible values are: **0b0** Enables transient load prediction. This is the default value. **0b1** Disables transient load prediction.	0b0
[27]	TLD_PRED_MODE	Aggressive Transient Load Prediction. The possible values are: **0b0** Disables aggressive transient load prediction. This is the default value. **0b1** Enables aggressive transient load prediction.	0b0
[26]	DTLB_CABT_EN	Enables TLB Conflict Data Abort Exception. The possible values are: **0b0** Disables TLB conflict data abort exception. This is the default value. **0b1** Enables TLB conflict data abort exception.	0b0
[25:24]	WS_THR_L2	Threshold for direct stream to L2 cache on store. The possible values are: **0b00** 256B - This is the default value **0b01** 4KB **0b10** 8KB **0b11** Disables direct stream to L2 cache on store.	0b00
[23:22]	WS_THR_L3	Threshold for direct stream to L3 cache on store. The possible values are: **0b00** 128KB **0b01** 256KB - This is the default value **0b10** 512KB **0b11** Disables direct stream to L3 cache on store.	0b01
[21:20]	WS_THR_L4	Threshold for direct stream to L4 cache on store. The possible values are: **0b00** 256KB **0b01** 512KB - This is the default value **0b10** 1MB **0b11** Disables direct stream to L4 cache on store.	0b01

Bits	Name	Description	Reset
[19:18]	WS_THR_DRAM	Threshold for direct stream to DRAM on store. The possible values are: **0b00** 512KB **0b01** 1MB - This is the default value **0b10** 2MB **0b11** Disables direct stream to DRAM on store.	0b01
[17:16]	**RES0**	Reserved	**RES0**
[15]	PF_DIS	Disables hardware prefetching. The possible values are: **0b0** Enables hardware prefetching. This is the default value. **0b1** Disables hardware prefetching.	0b0
[14]	**RES0**	Reserved	**RES0**
[13:12]	PF_SS_L2_DIST	Single cache line stride prefetching L2 distance. The possible values are: **0b00** 22 lines ahead **0b01** 40 lines ahead **0b10** 60 lines ahead **0b11** Dynamic. This is the default value.	0b11
[11:10]	**RES0**	Reserved	**RES0**
[9]	PF_STS_DIS	Disable store-stride prefetches. The possible values are: **0b0** Enables store prefetching. This is the default value. **0b1** Disables store prefetching.	0b0
[8]	PF_STI_DIS	Disable store prefetches at issue (not overridden by ls_hw_pref_disable). The possible values are: **0b0** Enables store prefetching. This is the default value. **0b1** Disable store prefetching.	0b0
[7:6]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[5:4]	RPF_MODE	Region prefetcher aggressiveness. The possible values are: **0b00** Dynamic region prefetch aggressiveness. This is the default value. **0b01** Conservative region prefetching. **0b10** Very Conservative region prefetching. **0b11** Most Conservative region prefetching. This will disable the region prefetcher.	0b00
[3]	RPF_LO_CONF	Region Prefetcher single accesses training behavior. The possible values are: **0b0** Mostly don't train PHT on single access. This is the default value. **0b1** Always train the PHT on single access. This results in fewer prefetch requests.	0b0
[2:1]	**RES0**	Reserved	**RES0**
[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.	0b0

Access

MRS <Xt>, S3_0_C15_C1_4

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

MSR S3_0_C15_C1_4, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_4

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

```
        X[t, 64] = IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUECTLR_EL1;
```

MSR S3_0_C15_C1_4, <Xt>

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

A.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 xxx0 0001 1000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-11: AArch64_imp_cpuectlr2_el1 bit assignments

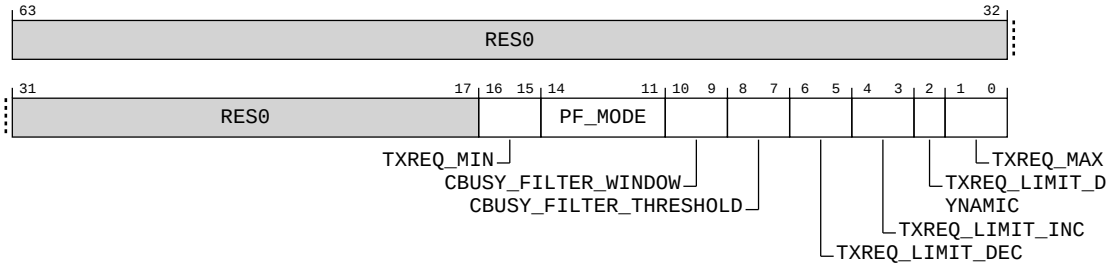


Table A-37: IMP_CPUECTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:17]	**RES0**	Reserved	**RES0**
[16:15]	TXREQ_MIN	Minimum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are: **0b00** 1/4 of L2 TQ size - This is the default value **0b01** 1/8 of L2 TQ size **0b10** 1/16 of L2 TQ size **0b11** 1/32 of L2 TQ size	0b00

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

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

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

Access

MRS <Xt>, S3_0_C15_C1_5

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

MSR S3_0_C15_C1_5, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_5

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

MSR S3_0_C15_C1_5, <Xt>

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

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

A.1.12 IMP_CPUBUSQOS_EL1, CPU Bus QoS 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 1110 1010 1110
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-12: AArch64_imp_cpubusqos_el1 bit assignments

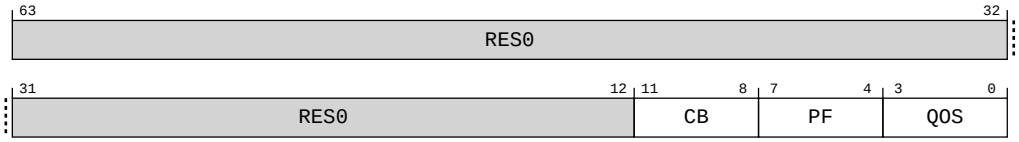


Table A-40: IMP_CPUBUSQOS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	RES0
[11:8]	CB	Value driven on TXREQ Qos for copyback transactions.	0b1110
[7:4]	PF	Value driven on TXREQ QoS for prefetch accesses.	0b1010
[3:0]	QOS	Value driven on TXREQ QoS field for demand accesses.	0b1110

Access

MRS <Xt>, S3_0_C15_C1_7

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

MSR S3_0_C15_C1_7, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C1_7

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUBUSQOS_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUBUSQOS_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUBUSQOS_EL1;
```

MSR S3_0_C15_C1_7, <Xt>

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

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.QOSEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUBUSQOS_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.QOSEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUBUSQOS_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUBUSQOS_EL1 = X[t, 64];
```

A.1.13 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-13: AArch64_imp_cpul2dirtylnct_el1 bit assignments

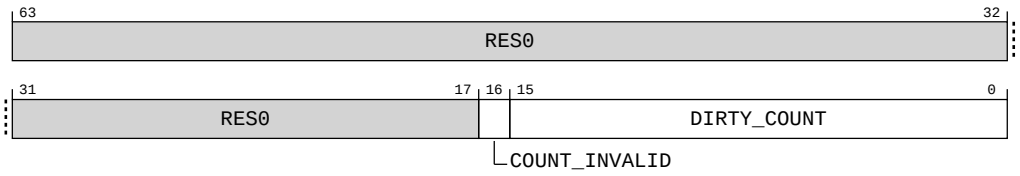


Table A-43: IMP_CPUL2DIRTYLNCT_EL1 bit descriptions

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.1.14 IMP_CPUPWRCTLR_EL1, CPU Power Control Register

This register controls various power aspects of the core.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 0000 xxx0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-14: AArch64_imp_cpupwrctlr_el1 bit assignments

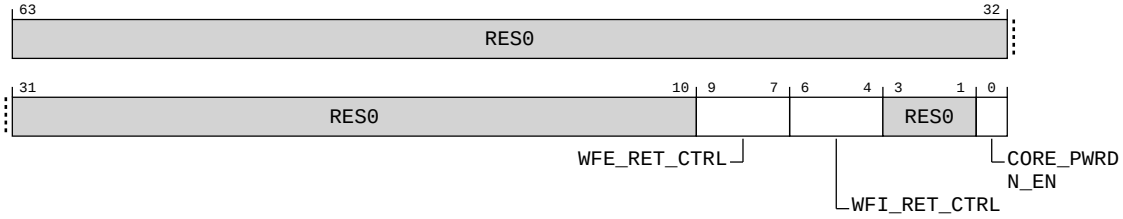


Table A-45: IMP_CPUPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:10]	RES0	Reserved	RES0
[9:7]	WFE_RET_CTRL	Wait for Event retention control. The possible values are: **0b000** Dynamic retention is disabled. **0b001** 2 system counter ticks are required before retention entry. **0b010** 8 system counter ticks are required before retention entry. **0b011** 32 system counter ticks are required before retention entry. **0b100** 64 system counter ticks are required before retention entry. **0b101** 128 system counter ticks are required before retention entry. **0b110** 256 system counter ticks are required before retention entry. **0b111** 512 system counter ticks are required before retention entry.	0b000

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

Access

MRS <Xt>, S3_0_C15_C2_7

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

MSR S3_0_C15_C2_7, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C2_7

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

```
        X[t, 64] = IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPWRCTLR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPWRCTLR_EL1;
```

MSR S3_0_C15_C2_7, <Xt>

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

A.1.15 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register (EL1)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-15: AArch64_imp_atcr_el1 bit assignments

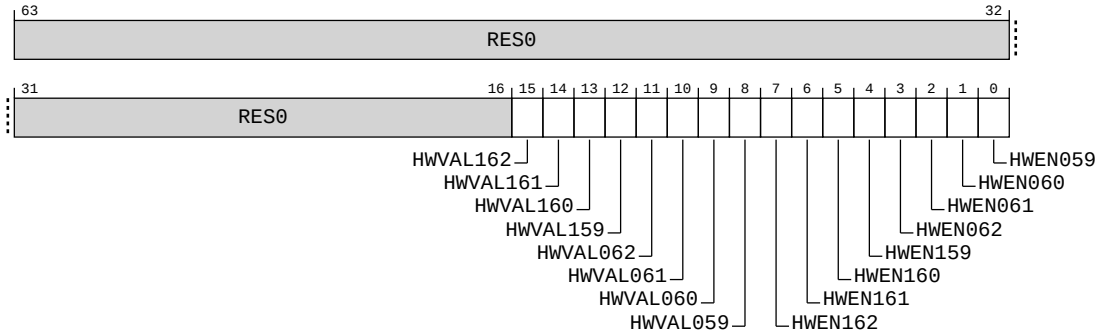


Table A-48: IMP_ATCR_EL1 bit descriptions

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

Access

MRS <Xt>, S3_0_C15_C7_0

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

MSR S3_0_C15_C7_0, <Xt>

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

Accessibility

MRS <Xt>, S3_0_C15_C7_0

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

MSR S3_0_C15_C7_0, <Xt>

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

A.1.16 IMP_CPUACTLR5_EL1, CPU Auxiliary Control Register 5 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-16: AArch64_imp_cpuactlr5_el1 bit assignments

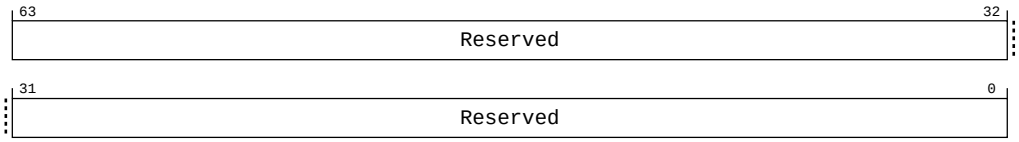


Table A-51: IMP_CPUACTLR5_EL1 bit descriptions

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.1.17 IMP_CPUACTLR6_EL1, CPU Auxiliary Control Register 6 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-17: AArch64_imp_cpuactlr6_el1 bit assignments

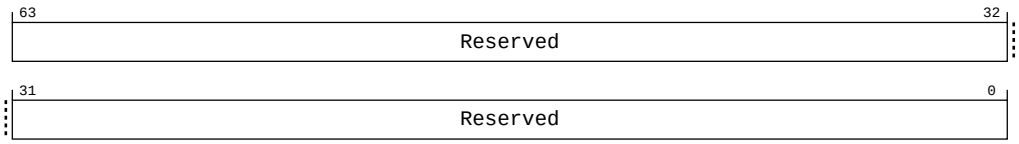


Table A-54: IMP_CPUACTLR6_EL1 bit descriptions

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.1.18 IMP_CPUACTLR7_EL1, CPU Auxiliary Control Register 7 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-18: AArch64_imp_cpuactlr7_el1 bit assignments

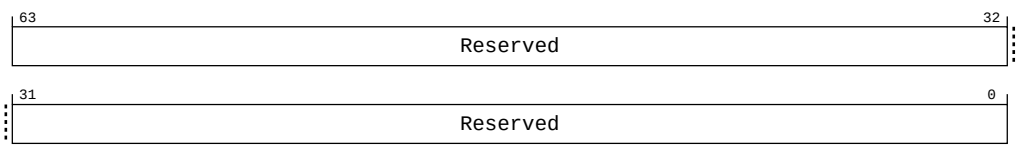


Table A-57: IMP_CPUACTLR7_EL1 bit descriptions

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.1.19 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-19: AArch64_imp_cpuactlr8_el1 bit assignments

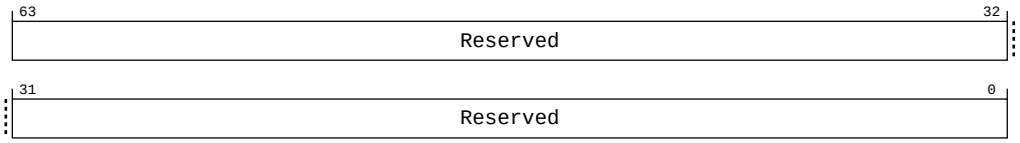


Table A-60: IMP_CPUACTLR8_EL1 bit descriptions

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.1.20 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 xxxx
```

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-20: AArch64_imp_cpuactlr9_el1 bit assignments

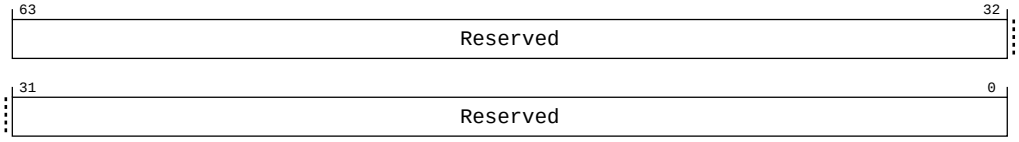


Table A-63: IMP_CPUACTLR9_EL1 bit descriptions

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.1.21 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

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0
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-21: AArch64_aidr_el1 bit assignments

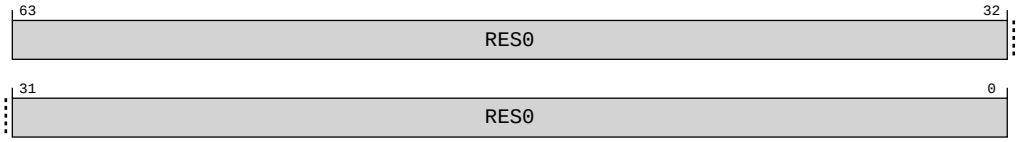


Table A-66: AIDR_EL1 bit descriptions

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

Access

MRS <Xt>, AIDR_EL1

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

Accessibility

MRS <Xt>, AIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = AIDR_EL1;
```

A.1.22 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-22: AArch64_fpcr bit assignments

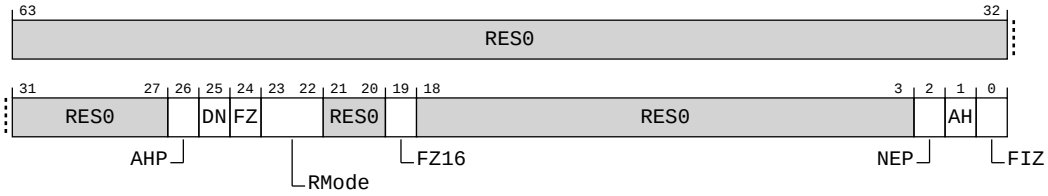


Table A-68: FPCR bit descriptions

Bits	Name	Description	Reset
[63:27]	RES0	Reserved	RES0
[26]	AHP	Alternative half-precision control bit. **0b0** IEEE half-precision format selected. **0b1** Alternative half-precision format selected.	x
[25]	DN	Default NaN use for NaN propagation. **0b0** NaN operands propagate through to the output of a floating-point operation. **0b1** Any operation involving one or more NaNs returns the Default NaN. This bit has no effect on the output of FABS, FMAX*, FMIN*, and FNEG instructions, and a default NaN is never returned as a result of these instructions.	x

Bits	Name	Description	Reset
[24]	FZ	Flushing denormalized numbers to zero control bit. **0b0** If FPCR.AH is 0, the flushing to zero of single-precision and double-precision denormalized inputs to, and outputs of, floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. If FPCR.AH is 1, the flushing to zero of single-precision and double-precision denormalized outputs of floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. **0b1** If FPCR.AH is 0, denormalized single-precision and double-precision inputs to, and outputs from, floating-point instructions are flushed to zero. If FPCR.AH is 1, denormalized single-precision and double-precision outputs from floating-point instructions are flushed to zero.	x
[23:22]	RMode	Rounding Mode control field. **0b00** Round to Nearest (RN) mode. **0b01** Round towards Plus Infinity (RP) mode. **0b10** Round towards Minus Infinity (RM) mode. **0b11** Round towards Zero (RZ) mode.	xx
[21:20]	**RES0**	Reserved	**RES0**
[19]	FZ16	Flushing denormalized numbers to zero control bit on half-precision data-processing instructions. **0b0** For some instructions, this bit disables flushing to zero of inputs and outputs that are half-precision denormalized numbers. **0b1** Flushing denormalized numbers to zero enabled. For some instructions that do not convert a half-precision input to a higher precision output, this bit enables flushing to zero of inputs and outputs that are half-precision denormalized numbers.	x
[18:3]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[2]	NEP	Controls how the output elements other than the lowest element of the vector are determined for Advanced SIMD scalar instructions. **0b0** Does not affect how the output elements other than the lowest are determined for Advanced SIMD scalar instructions. **0b1** The output elements other than the lowest are taken from the following registers: • For 3-input scalar versions of the FMLA (by element) and FMLS (by element) instructions, the <Hd>, <Sd>, or <Dd> register. • For 3-input versions of the FMADD, FMSUB, FNMADD, and FNMSUB instructions, the <Ha>, <Sa>, or <Da> register. • For 2-input scalar versions of the FACGE, FACGT, FCMEQ (register), FCMGE (register), and FCMGT (register) instructions, the <Hm>, <Sm>, or <Dm> register. • For 2-input scalar versions of the FABD, FADD (scalar), FDIV (scalar), FMAX (scalar), FMAXNM (scalar), FMIN (scalar), FMINNM (scalar), FMUL (by element), FMUL (scalar), FMULX (by element), FMULX, FNMUL (scalar), FRECPS, FRSQRTS, and FSUB (scalar) instructions, the <Hn>, <Sn>, or <Dn> register. • For 1-input scalar versions of the following instructions, the <Hd>, <Sd>, or <Dd> register: ◦ The (vector) versions of the FCVTAS, FCVTAU, FCVTMS, FCVTMU, FCVTNS, FCVTNU, FCVTPS, and FCVTPU instructions. ◦ The (vector, fixed-point) and (vector, integer) versions of the FCVTZS, FCVTZU, SCVTF, and UCVTF instructions. ◦ The (scalar) versions of the FABS, FNEG, FRINT32X, FRINT32Z, FRINT64X, FRINT64Z, FRINTA, FRINTI, FRINTM, FRINTN, FRINTP, FRINTX, FRINTZ, and FSQRT instructions. ◦ The (scalar, fixed-point) and (scalar, integer) versions of the SCVTF and UCVTF instructions. ◦ The BFCVT, FCVT, FCVTXN, FRECPE, FRECPX, and FRSQRTE instructions.	x
[1]	AH	Alternate Handling. Controls alternate handling of floating-point numbers. **0b1**	x
[0]	FIZ	Flush Inputs to Zero. Controls whether single-precision, double-precision and BFloat16 input operands that are denormalized numbers are flushed to zero. **0b0** The flushing to zero of single-precision and double-precision denormalized inputs to floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. **0b1** Denormalized single-precision and double-precision inputs to most floating-point instructions flushed to zero.	x

Access

MRS <Xt>, FPCR

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

MSR FPCR, <Xt>

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

Accessibility

MRS <Xt>, FPCR

```
if PSTATE.EL == EL0 then
    if !(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.1.23 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 000x 0xxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-23: AArch64_actlr_el2 bit assignments

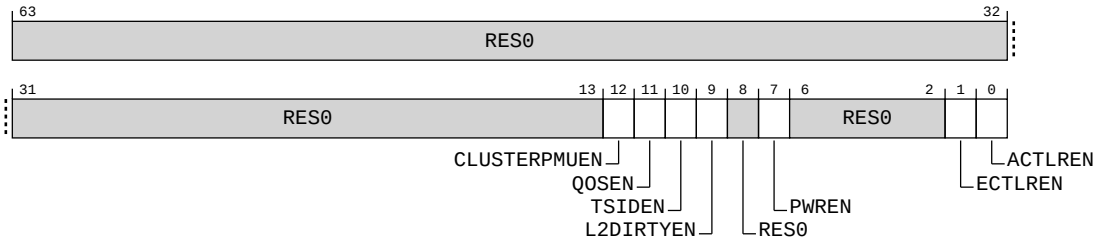


Table A-71: ACTLR_EL2 bit descriptions

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
[11]	QOSEN	CPU Bus QoS Register enable. The possible values are: **0b0** Register CPUBUSQOS_EL1 is not write-accessible from EL1. This is the reset value. **0b1** Register CPUBOSQOS_EL1 is write-accessible from EL1 if it is also write-accessible from EL2	0b0

Bits	Name	Description	Reset
[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]	**RES0**	Reserved	**RES0**
[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
[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

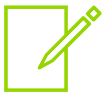
```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ACTLR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACTLR_EL2;
```

MSR ACTLR_EL2, <Xt>

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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-24: AArch64_hacr_el2 bit assignments

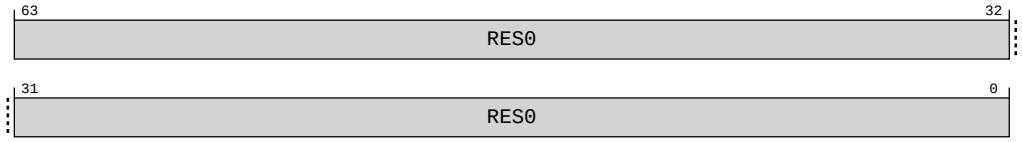


Table A-74: HACR_EL2 bit descriptions

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
    if EL2Enabled() && HCR_EL2.NV == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = HACR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = HACR_EL2;
```

MSR HACR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HACR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    HACR_EL2 = X[t, 64];
```

A.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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-25: AArch64_afsr0_el2 bit assignments

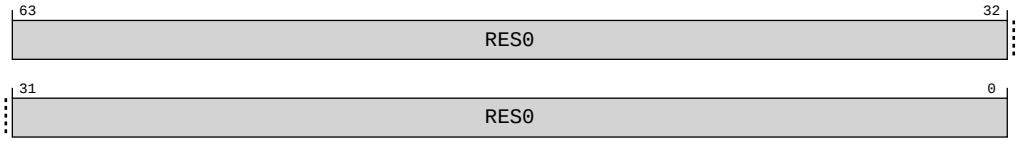


Table A-77: AFSR0_EL2 bit descriptions

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

MSR AFSR0_EL2, <Xt>

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

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x128];
    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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR0_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t, 64];
    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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-26: AArch64_afsr1_el2 bit assignments

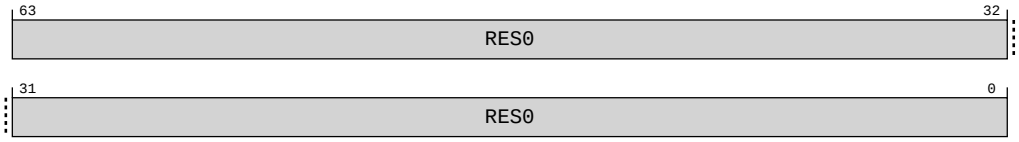


Table A-82: AFSR1_EL2 bit descriptions

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

MSR AFSR1_EL2, <Xt>

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

```
elsif PSTATE.EL == EL2 then
    AFSR1_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    AFSR1_EL2 = X[t, 64];
```

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x130];
    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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x130] = X[t, 64];
    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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-27: AArch64_amair_el2 bit assignments

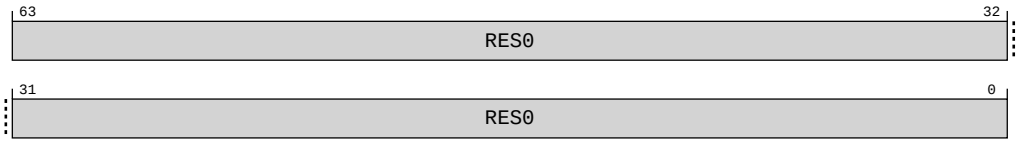


Table A-87: AMAIR_EL2 bit descriptions

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

MSR AMAIR_EL2, <Xt>

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

MRS <Xt>, AMAIR_EL1

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

```
        X[t, 64] = NVMem[0x148];
    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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x148] = X[t, 64];
    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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-28: AArch64_imp_atcr_el2 bit assignments

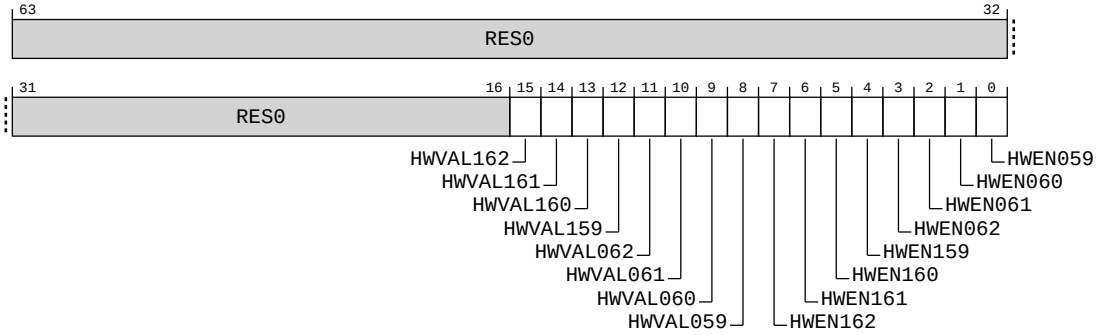


Table A-92: IMP_ATCR_EL2 bit descriptions

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

Bits	Name	Description	Reset
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

MRS <Xt>, S3_4_C15_C7_0

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b000

MSR S3_4_C15_C7_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_4_C15_C7_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_ATCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL2;
```

MSR S3_4_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL2 = X[t, 64];
```

A.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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-29: AArch64_imp_avtcr_el2 bit assignments

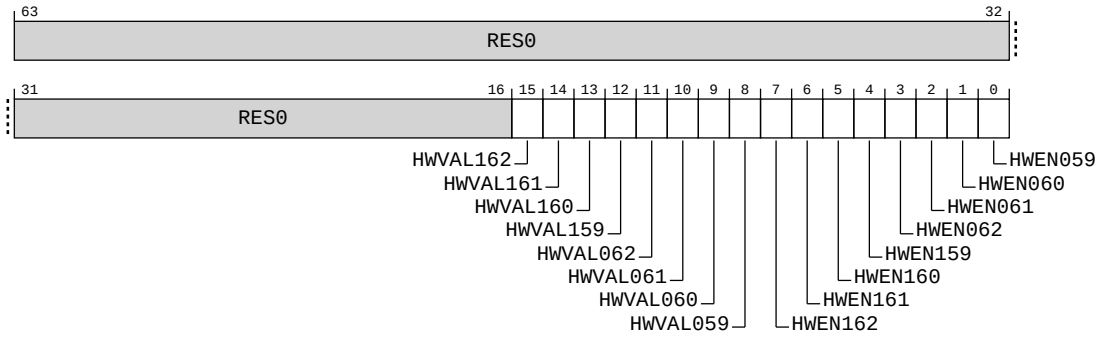


Table A-95: IMP_AVTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN162 is set.	0b0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN161 is set.	0b0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN160 is set.	0b0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN159 is set.	0b0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN062 is set.	0b0

Bits	Name	Description	Reset
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN059 is set.	0b0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

MRS <Xt>, S3_4_C15_C7_1

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

MSR S3_4_C15_C7_1, <Xt>

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

Accessibility

MRS <Xt>, S3_4_C15_C7_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_AVTCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_AVTCR_EL2;
```

MSR S3_4_C15_C7_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_AVTCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_AVTCR_EL2 = X[t, 64];
```

A.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 000x 0xxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-30: AArch64_actlr_el3 bit assignments

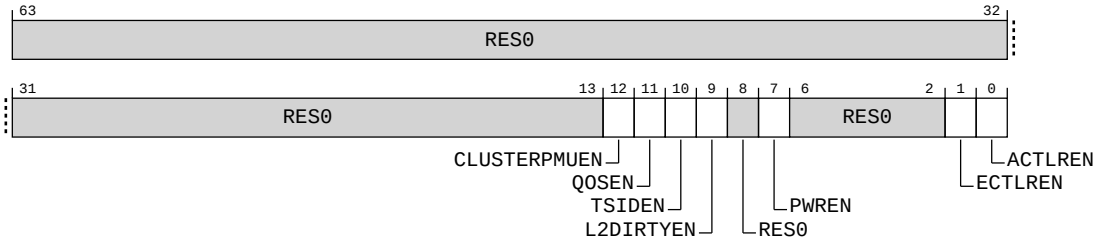


Table A-98: ACTLR_EL3 bit descriptions

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]	QOSEN	CPU Bus QoS Register enable. The possible values are: **0b0** Register CPUBUSQOS_EL1 is not write-accessible EL2 and EL1. This is the reset value. **0b1** Register CPUBOSQOS_EL1 is write-accessible from EL2, and EL1 (if 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
[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]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[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 GPTBR_EL3, Granule Protection Table Base Register

The control register for Granule Protection Table base address.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-31: AArch64_gptbr_el3 bit assignments

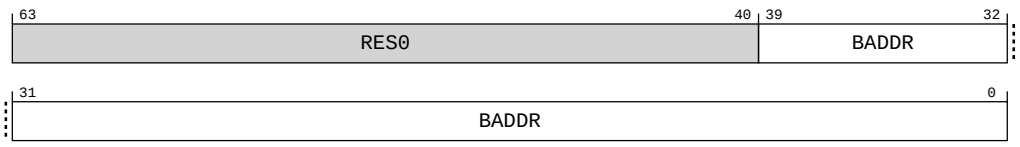


Table A-101: GPTBR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:40]	**RES0**	Reserved	**RES0**
[39:0]	BADDR	Base address for the level 0 GPT. This field represents bits [51:12] of the level 0 GPT base address.	`40{x}`

Access

MRS <Xt>, GPTBR_EL3

op0	op1	CRn	CRm	op2
`0b11`	`0b110`	`0b0010`	`0b0001`	`0b100`

MSR GPTBR_EL3, <Xt>

op0	op1	CRn	CRm	op2
`0b11`	`0b110`	`0b0010`	`0b0001`	`0b100`

Accessibility

MRS <Xt>, GPTBR_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] = GPTBR_EL3;
```

MSR GPTBR_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
    GPTBR_EL3 = X[t, 64];
```

A.1.32 GPCCR_EL3, Granule Protection Check Control Register (EL3)

The control register for Granule Protection Checks.

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 xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-32: AArch64_gpccr_el3 bit assignments

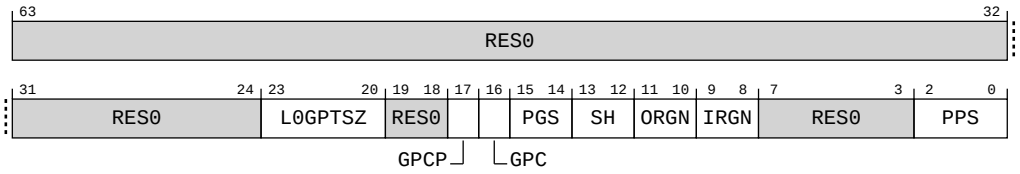


Table A-104: GPCCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**
[23:20]	L0GPTSZ	Level 0 GPT entry size. This field advertises the number of least-significant address bits protected by each entry in the level 0 GPT. **0b0000** 30-bits. Each entry covers 1GB of address space. **0b0100** 34-bits. Each entry covers 16GB of address space. **0b0110** 36-bits. Each entry covers 64GB of address space. **0b1001** 39-bits. Each entry covers 512GB of address space. Access to this field is: RO	xxxx

Bits	Name	Description	Reset
[19:18]	**RES0**	Reserved	**RES0**
[17]	GPCP	Granule Protection Check Priority. This control governs behavior of granule protection checks on fetches of stage 2 Table descriptors. **0b0** GPC faults are all reported with a priority that is consistent with the GPC being performed on any access to physical address space. **0b1** A GPC fault for the fetch of a Table descriptor for a stage 2 translation table walk might not be generated or reported. All other GPC faults are reported with a priority consistent with the GPC being performed on all accesses to physical address spaces.	x
[16]	GPC	Granule Protection Check Enable. **0b0** Granule protection checks are disabled. Accesses are not prevented by this mechanism. **0b1** All accesses to physical address spaces are subject to granule protection checks, except for fetches of GPT information and accesses governed by the GPCCR_EL3.GPCP control.	0b0
[15:14]	PGS	Physical Granule size. **0b00** 4KB. **0b01** 64KB. **0b10** 16KB.	xx
[13:12]	SH	GPT fetch Shareability attribute **0b00** Non-shareable. **0b10** Outer Shareable. **0b11** Inner Shareable.	xx
[11:10]	ORGN	GPT fetch Outer cacheability attribute. **0b00** Normal memory, Outer Non-cacheable. **0b01** Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable. **0b10** Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable. **0b11** Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.	xx

Bits	Name	Description	Reset
[9:8]	IRGN	GPT fetch Inner cacheability attribute. **0b00** Normal memory, Inner Non-cacheable. **0b01** Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable. **0b10** Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable. **0b11** Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.	xx
[7:3]	**RES0**	Reserved	**RES0**
[2:0]	PPS	Protected Physical Address Size. The size of the memory region protected by AArch64-GPTBR_EL3, in terms of the number of least-significant address bits. **0b000** 32 bits, 4GB protected address space. **0b001** 36 bits, 64GB protected address space. **0b010** 40 bits, 1TB protected address space. **0b011** 42 bits, 4TB protected address space. **0b100** 44 bits, 16TB protected address space. **0b101** 48 bits, 256TB protected address space. **0b110** 52 bits, 4PB protected address space.	xxx

Access

MRS <Xt>, GPCCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b110

MSR GPCCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0010	0b0001	0b110

Accessibility

MRS <Xt>, GPCCR_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] = GPCCR_EL3;
```

MSR GPCCR_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
    GPCCR_EL3 = X[t, 64];
```

A.1.33 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

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0
xxxx	xxxx	xxxx	xxxx	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-33: AArch64_afsr0_el3 bit assignments

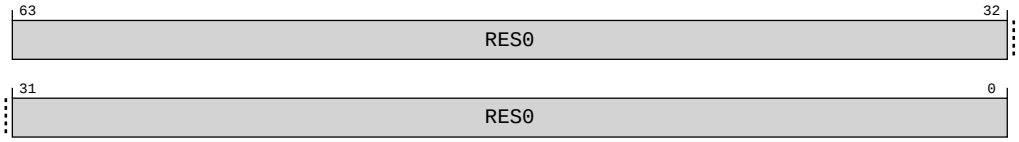


Table A-107: AFSR0_EL3 bit descriptions

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.1.34 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-34: AArch64_afsr1_el3 bit assignments

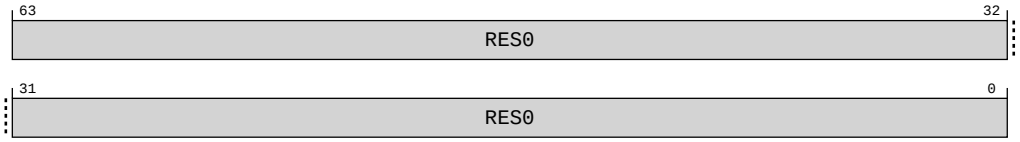


Table A-110: AFSR1_EL3 bit descriptions

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.1.35 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-35: AArch64_amair_el3 bit assignments

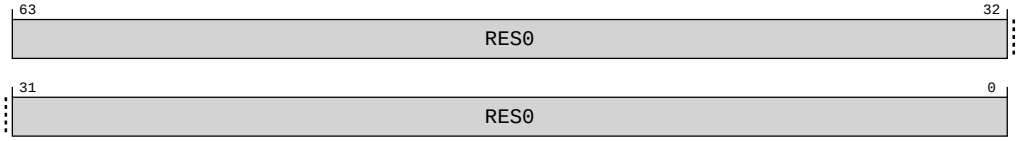


Table A-113: AMAIR_EL3 bit descriptions

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.1.36 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-36: AArch64_rmr_el3 bit assignments

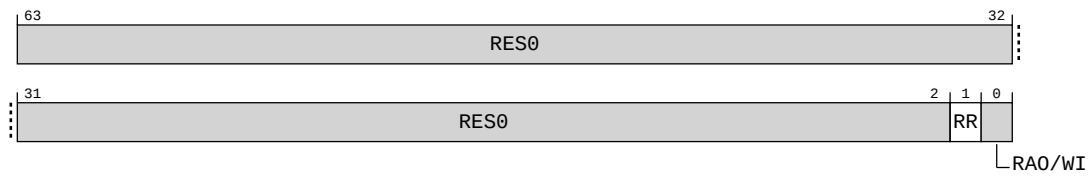


Table A-116: RMR_EL3 bit descriptions

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.1.37 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-37: AArch64_imp_cpul2sdirtylnct_el3 bit assignments

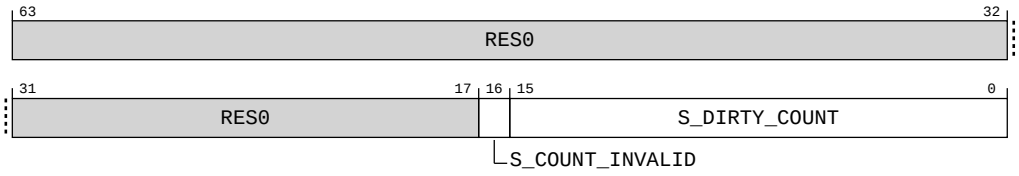


Table A-119: IMP_CPUL2SDIRTYLNCT_EL3 bit descriptions

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.1.38 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-38: AArch64_imp_cpuactlr_el3 bit assignments

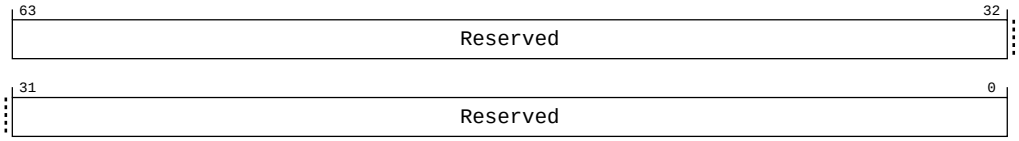


Table A-121: IMP_CPUACTLR_EL3 bit descriptions

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.1.39 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register (EL3)

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 xxxx 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-39: AArch64_imp_atcr_el3 bit assignments

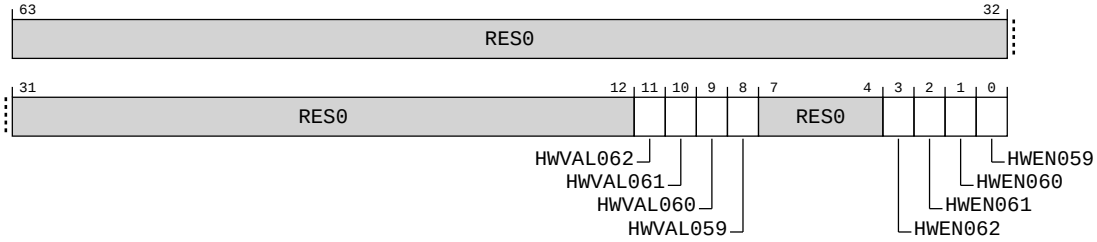


Table A-124: IMP_ATCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	**RES0**	Reserved	**RES0**
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN062 is set.	0b0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN061 is set.	0b0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN060 is set.	0b0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN059 is set.	0b0
[7:4]	**RES0**	Reserved	**RES0**
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

Access

MRS <Xt>, S3_6_C15_C7_0

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

MSR S3_6_C15_C7_0, <Xt>

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

Accessibility

MRS <Xt>, S3_6_C15_C7_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_ATCR_EL3;
```

MSR S3_6_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t, 64];
```

A.1.40 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-40: AArch64_imp_cpupselr_el3 bit assignments

Table A-127: 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.1.41 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-41: AArch64_imp_cpupcr_el3 bit assignments

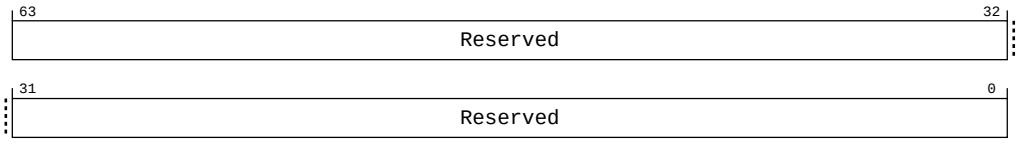


Table A-130: IMP_CPUPCR_EL3 bit descriptions

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.1.42 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
```

```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-42: AArch64_imp_cpupor_el3 bit assignments

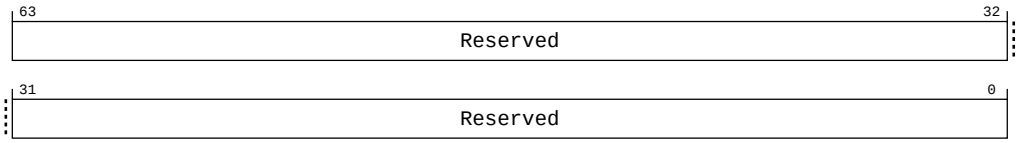


Table A-133: IMP_CPUPOR_EL3 bit descriptions

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.1.43 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-43: AArch64_imp_cpupmr_el3 bit assignments

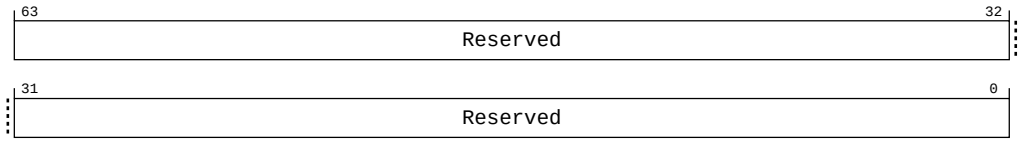


Table A-136: IMP_CPUPMR_EL3 bit descriptions

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.1.44 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-44: AArch64_imp_cpupor2_el3 bit assignments

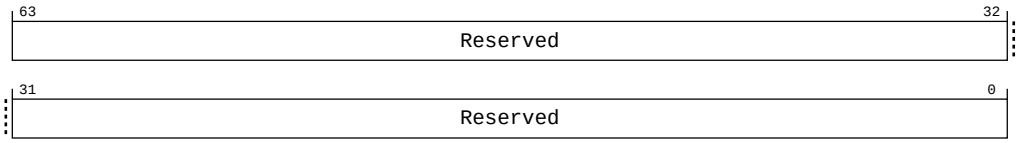


Table A-139: IMP_CPUPOR2_EL3 bit descriptions

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.1.45 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-45: AArch64_imp_cpupmr2_el3 bit assignments

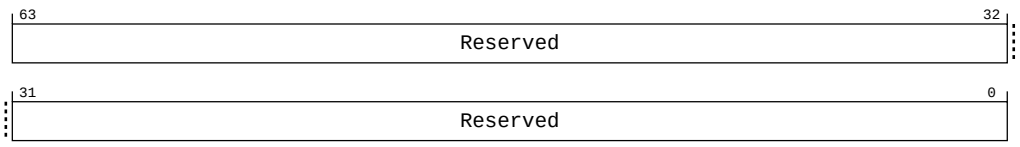


Table A-142: IMP_CPUPMR2_EL3 bit descriptions

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.1.46 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-46: AArch64_imp_cpupfr_el3 bit assignments

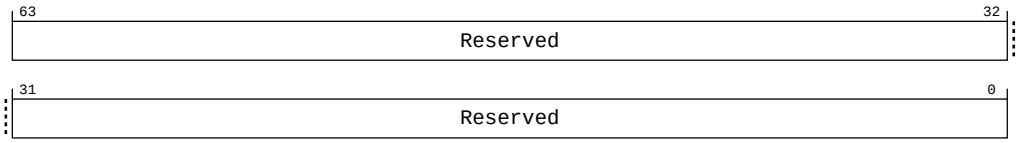


Table A-145: IMP_CPUPFR_EL3 bit descriptions

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 registers summary

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

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

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

Table A-148: registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPUPPMPDPCR_EL1	3	0	C15	C2	4	See individual bit resets.	64-bit	Performance and Power Management PDP Control Register
IMP_CPUPPMCR_EL3	3	6	C15	C2	0	See individual bit resets.	64-bit	Global Performance and Power Management Configuration Register
IMP_CPUMPMMCR_EL3	3	6	C15	C2	1	See individual bit resets.	64-bit	Global MPMM Control Register
IMP_CPUPPMCR4_EL3	3	6	C15	C2	4	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR5_EL3	3	6	C15	C2	5	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR6_EL3	3	6	C15	C2	6	See individual bit resets.	64-bit	CPU Power Performance Management Control Register

A.2.1 IMP_CPUPPMPDPCR_EL1, Performance and Power Management PDP Control Register

Provides **IMPLEMENTATION DEFINED** control of the Performance Defined Power (PDP) feature.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-47: AArch64_imp_cpuppmpdpcr_el1 bit assignments

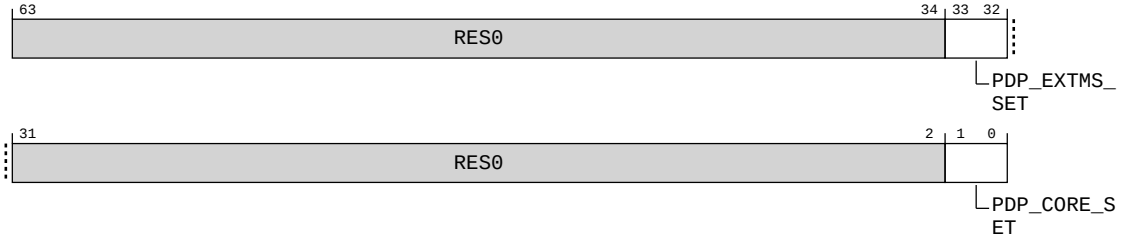


Table A-149: IMP_CPUPPMPDPCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:34]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[33:32]	PDP_EXTMS_SET	External memory system PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00
[31:2]	**RES0**	Reserved	**RES0**
[1:0]	PDP_CORE_SET	Core PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00

Access

MRS <Xt>, S3_0_C15_C2_4

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b100

MSR S3_0_C15_C2_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0010	0b100

Accessibility

MRS <Xt>, S3_0_C15_C2_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPPMPDPCR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPPMPDPCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMPDPCR_EL1;
```

MSR S3_0_C15_C2_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPPMPDPCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    IMP_CPUPPMPDPCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMPDPCR_EL1 = X[t, 64];
```

A.2.2 IMP_CPUPPMCR_EL3, Global Performance and Power Management Configuration Register

Provides **IMPLEMENTATION DEFINED** control and discovery of the Performance and Power Management (PPM) features.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x111 xxxx x011 xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-48: AArch64_imp_cpuppmcr_el3 bit assignments

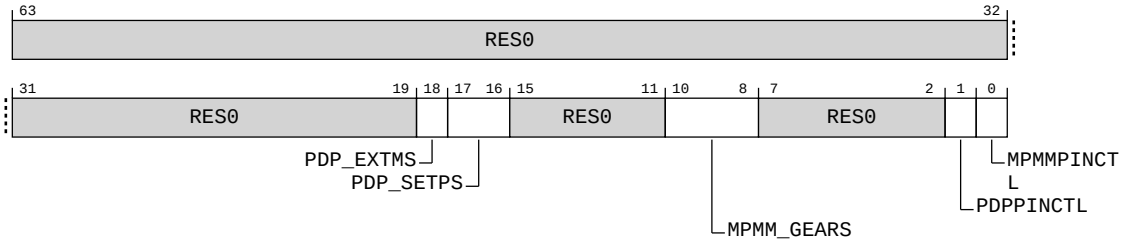


Table A-152: IMP_CPUPPMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	PDP_EXTMS	External memory system PDP control **0b1** Independent external memory system PDP control is implemented	0b1
[17:16]	PDP_SETPS	Number of PDP Setpoints Implemented **0b11** 3 PDP setpoints are implemented	0b11
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears Implemented **0b011** 3 MPMM gears are implemented	0b011
[7:2]	**RES0**	Reserved	**RES0**
[1]	PDPPINCTL	PDP Pin Control Enabled **0b0** PDP control through SPR and utility bus **0b1** PDP control through pin only	0b0
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus **0b1** MPMM control through pin only	0b0

Access

MRS <Xt>, S3_6_C15_C2_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b000

MSR S3_6_C15_C2_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b000

Accessibility

MRS <Xt>, S3_6_C15_C2_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR_EL3;
```

MSR S3_6_C15_C2_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t, 64];
```

A.2.3 IMP_CPUMPMMCR_EL3, Global MPMM Control Register

Provides **IMPLEMENTATION DEFINED** control of the Maximum Power Mitigation Mechanism (MPMM) feature.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-49: AArch64_imp_cpumpmmcr_el3 bit assignments

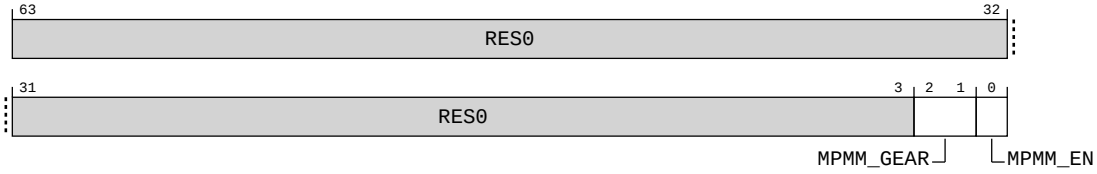


Table A-155: IMP_CPUMPMMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0 **0b01** Select MPMM Gear 1 **0b10** Select MPMM Gear 2 **0b11** Select MPMM Gear 3	0b00
[0]	MPMM_EN	MPMM Master Enable **0b0** MPMM is disabled **0b1** MPMM is enabled	0b0

Access

MRS <Xt>, S3_6_C15_C2_1

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b001

MSR S3_6_C15_C2_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b001

Accessibility

MRS <Xt>, S3_6_C15_C2_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUMPMMCR_EL3;
```

MSR S3_6_C15_C2_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUMPMMCR_EL3 = X[t, 64];
```

A.2.4 IMP_CPUPPMCR4_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-50: AArch64_imp_cpuppmcr4_el3 bit assignments

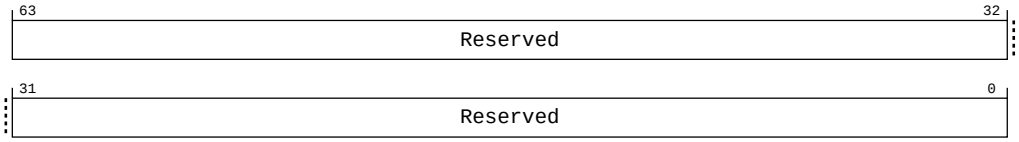


Table A-158: IMP_CPUPPMCR4_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_4

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b100

MSR S3_6_C15_C2_4, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b100

Accessibility

MRS <Xt>, S3_6_C15_C2_4

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR4_EL3;
```

MSR S3_6_C15_C2_4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR4_EL3 = X[t, 64];
```

A.2.5 IMP_CPUPPMCR5_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-51: AArch64_imp_cpuppmcr5_el3 bit assignments

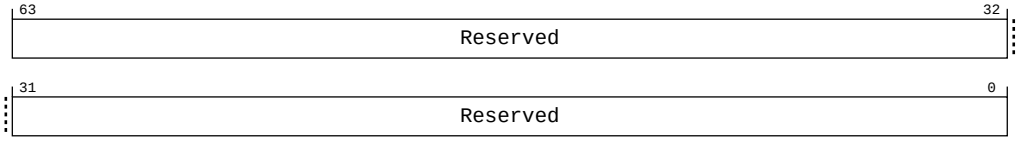


Table A-161: IMP_CPUPPMCR5_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_5

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b101

MSR S3_6_C15_C2_5, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b101

Accessibility

MRS <Xt>, S3_6_C15_C2_5

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR5_EL3;
```

MSR S3_6_C15_C2_5, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR5_EL3 = X[t, 64];
```

A.2.6 IMP_CPUPPMCR6_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-52: AArch64_imp_cpuppmcr6_el3 bit assignments

Table A-164: IMP_CPUPPMCR6_EL3 bit descriptions

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

Access

MRS <Xt>, S3_6_C15_C2_6

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b110

MSR S3_6_C15_C2_6, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0010	0b110

Accessibility

MRS <Xt>, S3_6_C15_C2_6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPUPPMCR6_EL3;
```

MSR S3_6_C15_C2_6, <Xt>

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

```
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR6_EL3 = X[t, 64];
```

A.3 AArch64 Debug registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-167: Debug registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
OSDTRRX_EL1	2	0	C0	C0	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Receive
DBGBVR0_EL1	2	0	C0	C0	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR0_EL1	2	0	C0	C0	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR0_EL1	2	0	C0	C0	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR0_EL1	2	0	C0	C0	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR1_EL1	2	0	C0	C1	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR1_EL1	2	0	C0	C1	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR1_EL1	2	0	C0	C1	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR1_EL1	2	0	C0	C1	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
MDCCINT_EL1	2	0	C0	C2	0	See individual bit resets.	64-bit	Monitor DCC Interrupt Enable Register
MDSCR_EL1	2	0	C0	C2	2	See individual bit resets.	64-bit	Monitor Debug System Control Register
DBGBVR2_EL1	2	0	C0	C2	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR2_EL1	2	0	C0	C2	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR2_EL1	2	0	C0	C2	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR2_EL1	2	0	C0	C2	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
OSDTRTX_EL1	2	0	C0	C3	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Transmit
DBGBVR3_EL1	2	0	C0	C3	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR3_EL1	2	0	C0	C3	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR3_EL1	2	0	C0	C3	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR3_EL1	2	0	C0	C3	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
DBGBVR4_EL1	2	0	C0	C4	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR4_EL1	2	0	C0	C4	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGBVR5_EL1	2	0	C0	C5	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR5_EL1	2	0	C0	C5	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers

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

A.3.1 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-53: AArch64_dbgbvr0_el1 bit assignments

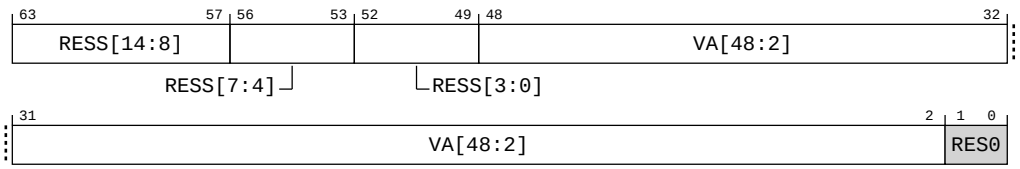


Table A-168: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-54: AArch64_dbgbvr0_el1 bit assignments

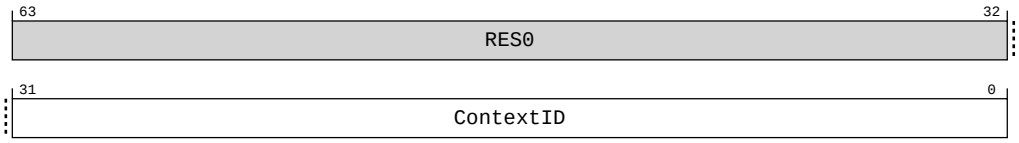


Table A-169: DBGBVR0_EL1 bit descriptions

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-55: AArch64_dbgbvr0_el1 bit assignments

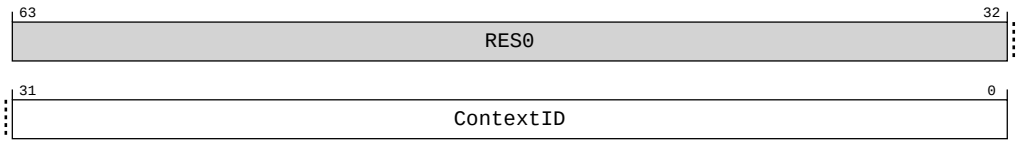


Table A-170: DBGBVR0_EL1 bit descriptions

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-56: AArch64_dbgbvr0_el1 bit assignments

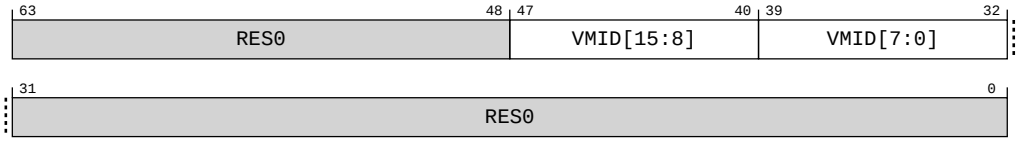


Table A-171: DBGBVR0_EL1 bit descriptions

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-57: AArch64_dbgbvr0_el1 bit assignments

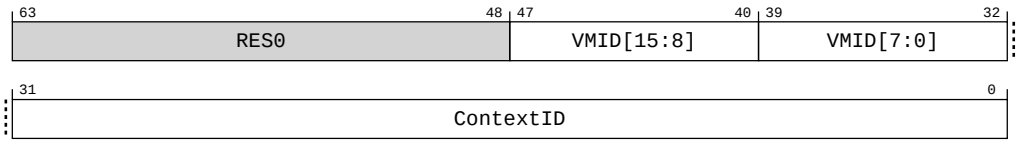


Table A-172: DBGBVR0_EL1 bit descriptions

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-58: AArch64_dbgbvr0_el1 bit assignments

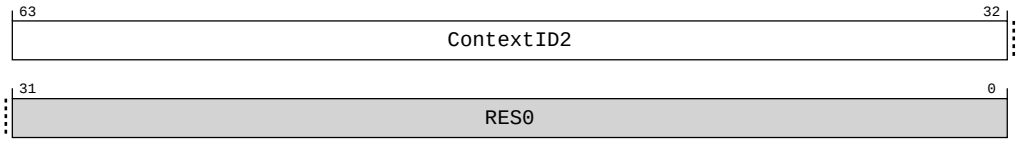


Table A-173: DBGBVR0_EL1 bit descriptions

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-59: AArch64_dbgbvr0_el1 bit assignments

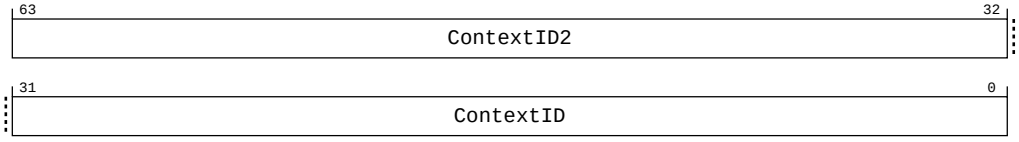


Table A-174: DBGBVR0_EL1 bit descriptions

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.2 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-60: AArch64_dbgbcr0_el1 bit assignments

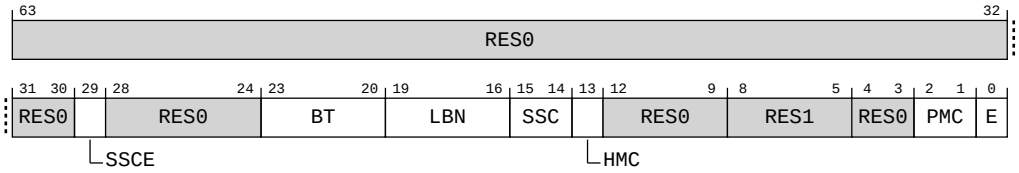


Table A-177: DBGBCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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

Bits	Name	Description	Reset
[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, SSC, and SSCE. 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, SSC, and SSCE. 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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.3 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-61: AArch64_dbgwvr0_el1 bit assignments

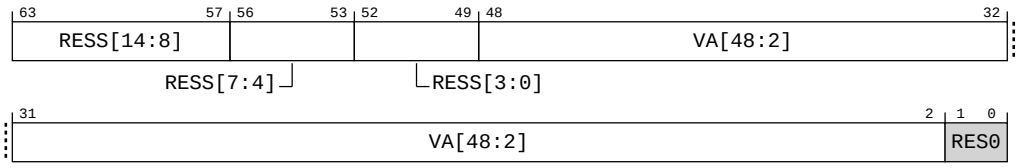


Table A-180: DBGWVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.4 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-62: AArch64_dbgwcr0_el1 bit assignments

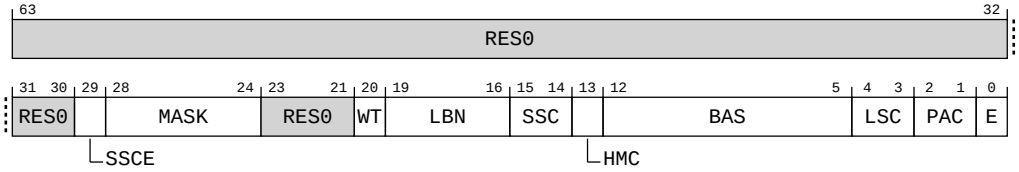


Table A-183: DBGWCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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, SSC, and SSCE. 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-184: BAS description on page 322 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-185: BAS description table 3 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 nonzero 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.	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, SSC, and SSCE. 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. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table A-184: BAS description

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-185: BAS description table 3

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.5 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

Where the reset reads xxxx, see individual bits.

Bit descriptions

When AArch64-DBGBCR1_EL1.BT == '000'

Figure A-63: AArch64_dbgbvr1_el1 bit assignments

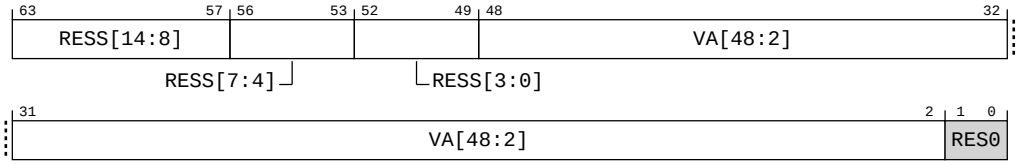


Table A-188: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-64: AArch64_dbgbvr1_el1 bit assignments

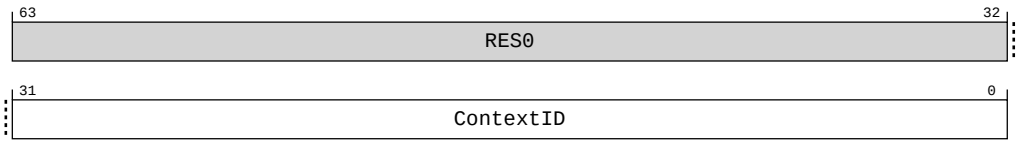


Table A-189: DBGBVR1_EL1 bit descriptions

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-65: AArch64_dbgbvr1_el1 bit assignments

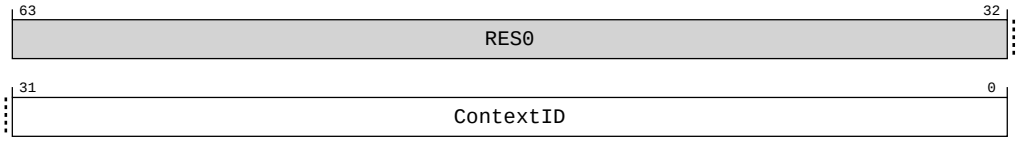


Table A-190: DBGBVR1_EL1 bit descriptions

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-66: AArch64_dbgbvr1_el1 bit assignments

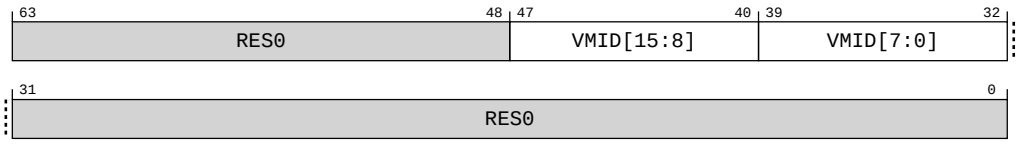


Table A-191: DBGBVR1_EL1 bit descriptions

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-67: AArch64_dbgbvr1_el1 bit assignments

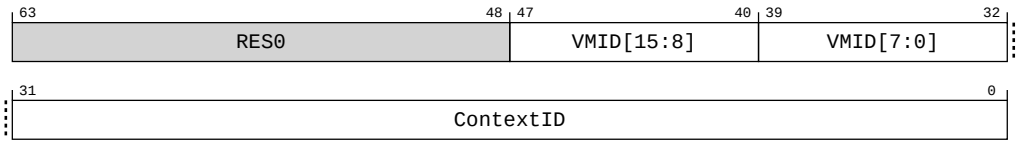


Table A-192: DBGBVR1_EL1 bit descriptions

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-68: AArch64_dbgbvr1_el1 bit assignments

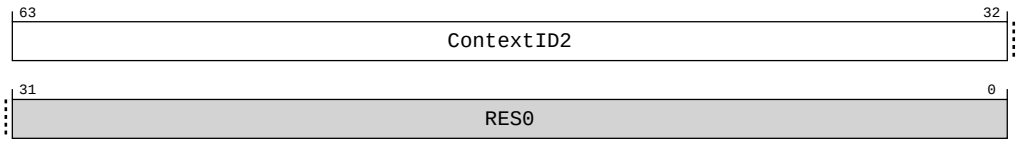


Table A-193: DBGBVR1_EL1 bit descriptions

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-69: AArch64_dbgbvr1_el1 bit assignments

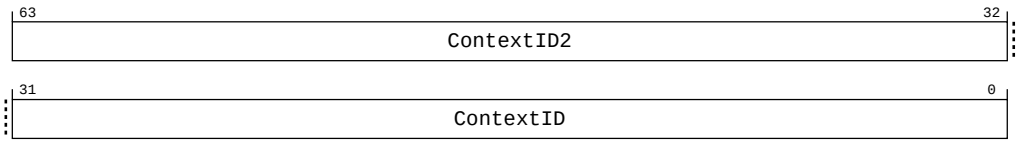


Table A-194: DBGBVR1_EL1 bit descriptions

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.6 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-70: AArch64_dbgbcr1_el1 bit assignments

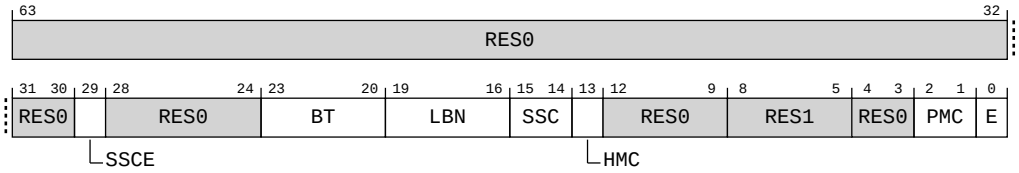


Table A-197: DBGBCR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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

Bits	Name	Description	Reset
[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, SSC, and SSCE. 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, SSC, and SSCE. 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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.7 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-71: AArch64_dbgwvr1_el1 bit assignments

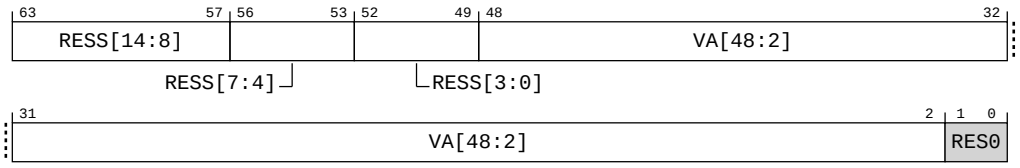


Table A-200: DBGWVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.8 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
```

```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-72: AArch64_dbgwcr1_el1 bit assignments

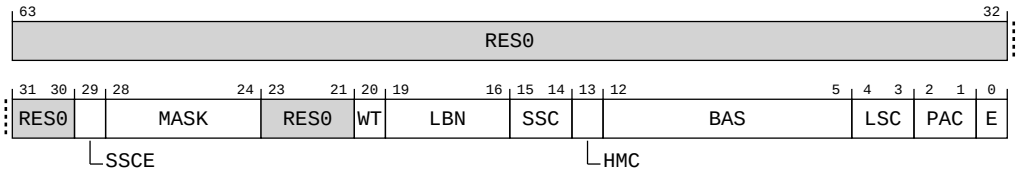


Table A-203: DBGWCR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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, SSC, and SSCE. 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-204: BAS description on page 338 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-205: BAS description table 3 on page 338 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 nonzero 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.	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, SSC, and SSCE. 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. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table A-204: BAS description

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-205: BAS description table 3

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.9 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-73: AArch64_dbgbvr2_el1 bit assignments

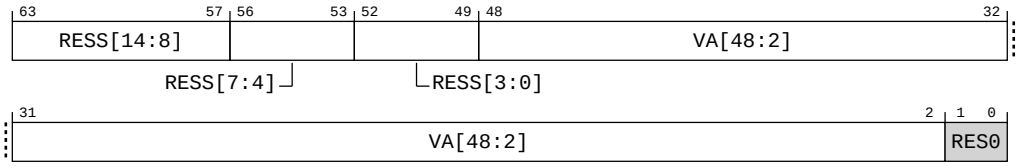


Table A-208: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-74: AArch64_dbgbvr2_el1 bit assignments

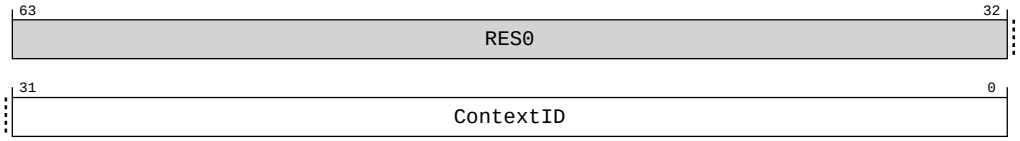


Table A-209: DBGBVR2_EL1 bit descriptions

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-75: AArch64_dbgbvr2_el1 bit assignments

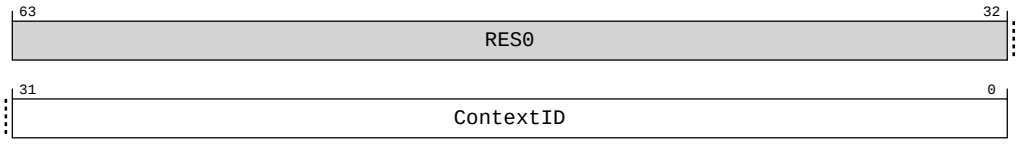


Table A-210: DBGBVR2_EL1 bit descriptions

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-76: AArch64_dbgbvr2_el1 bit assignments

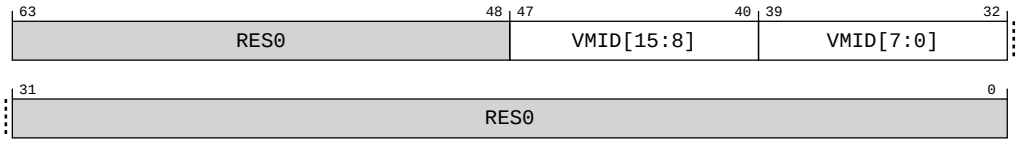


Table A-211: DBGBVR2_EL1 bit descriptions

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-77: AArch64_dbgbvr2_el1 bit assignments

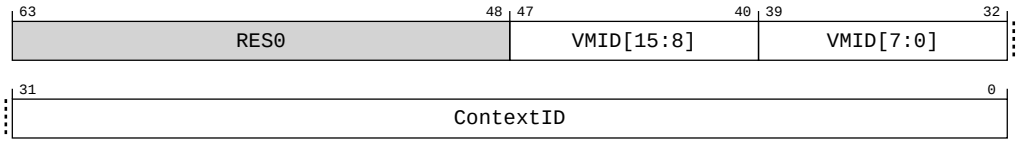


Table A-212: DBGBVR2_EL1 bit descriptions

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-78: AArch64_dbgbvr2_el1 bit assignments

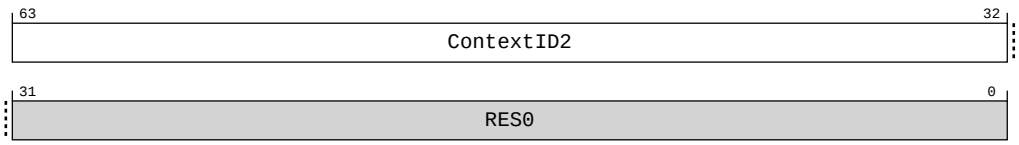


Table A-213: DBGBVR2_EL1 bit descriptions

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-79: AArch64_dbgbvr2_el1 bit assignments

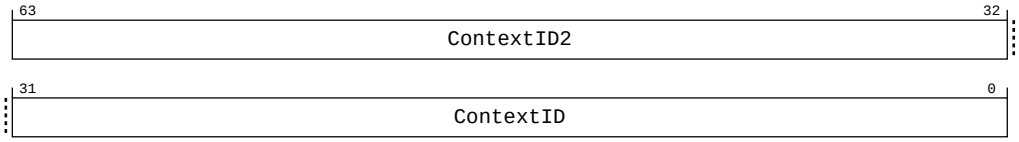


Table A-214: DBGBVR2_EL1 bit descriptions

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.10 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-80: AArch64_dbgbcr2_el1 bit assignments

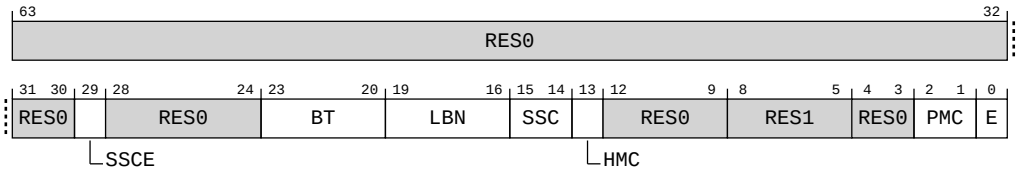


Table A-217: DBGBCR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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, SSC, and SSCE. 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, SSC, and SSCE. 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

Bits	Name	Description	Reset
[0]	E	Enable breakpoint n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.11 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
```

```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-81: AArch64_dbgwvr2_el1 bit assignments

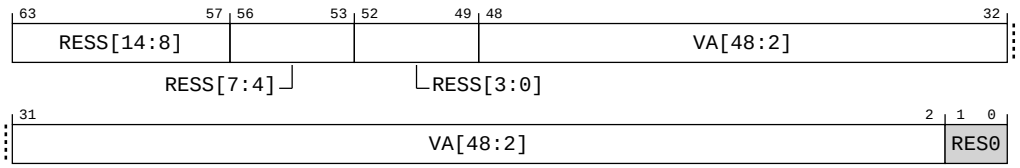


Table A-220: DBGWVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.12 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-82: AArch64_dbgwcr2_el1 bit assignments

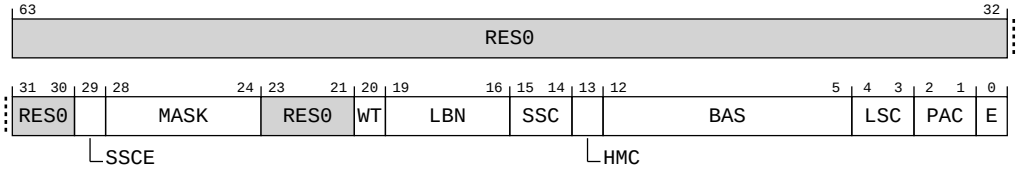


Table A-223: DBGWCR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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, SSC, and SSCE. 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

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 AArch64-DBGWVR<n>_EL1 is being watched. Table A-224: BAS description on page 354 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-225: BAS description table 3 on page 354 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 nonzero 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.	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, SSC, and SSCE. 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. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table A-224: BAS description

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-225: BAS description table 3

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

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.13 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-83: AArch64_dbgbvr3_el1 bit assignments

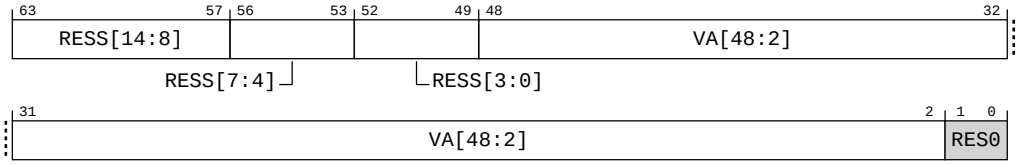


Table A-228: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-84: AArch64_dbgbvr3_el1 bit assignments

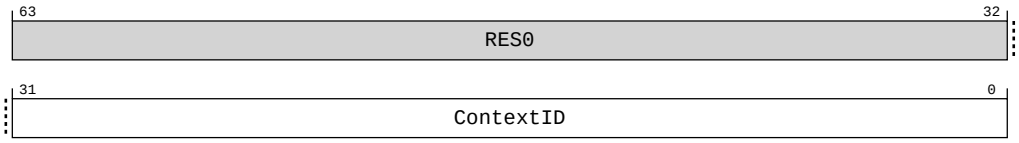


Table A-229: DBGBVR3_EL1 bit descriptions

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), 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-85: AArch64_dbgbvr3_el1 bit assignments

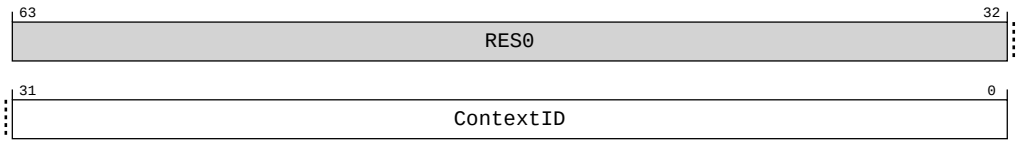


Table A-230: DBGBVR3_EL1 bit descriptions

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-86: AArch64_dbgbvr3_el1 bit assignments

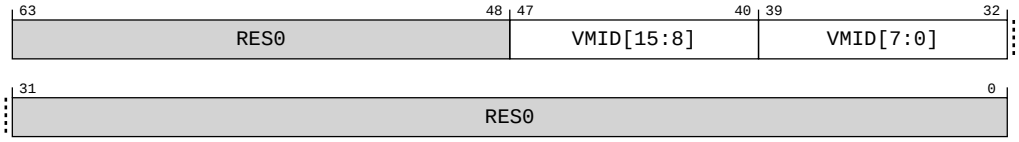


Table A-231: DBGBVR3_EL1 bit descriptions

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 AArch64-DBGBCR3_EL1.BT == '101'

Figure A-87: AArch64_dbgbvr3_el1 bit assignments

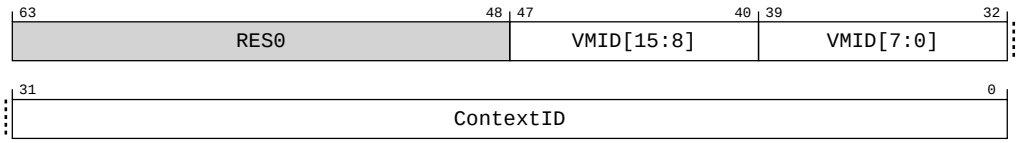


Table A-232: DBGBVR3_EL1 bit descriptions

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-DBGBCR3_EL1.BT == '110'

Figure A-88: AArch64_dbgbvr3_el1 bit assignments

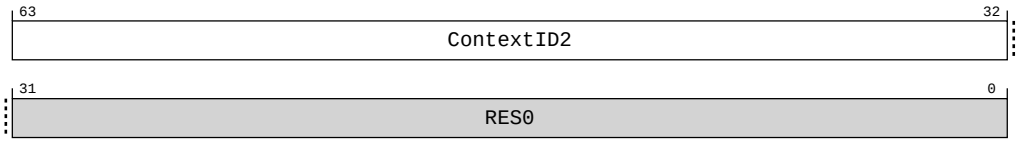


Table A-233: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	ContextID2	Context ID value for comparison against AArch64-CONTEXTIDR_EL2.	32{x}
[31:0]	RES0	Reserved	RES0

When AArch64-DBGBCR3_EL1.BT == '111'

Figure A-89: AArch64_dbgbvr3_el1 bit assignments

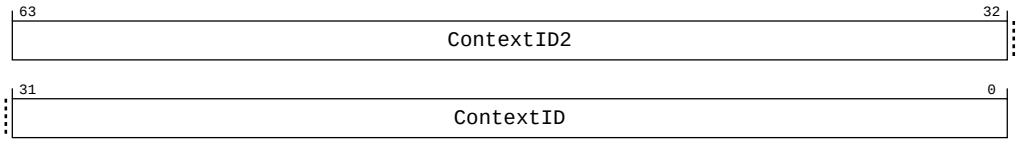


Table A-234: DBGBVR3_EL1 bit descriptions

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.14 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-90: AArch64_dbgbcr3_el1 bit assignments

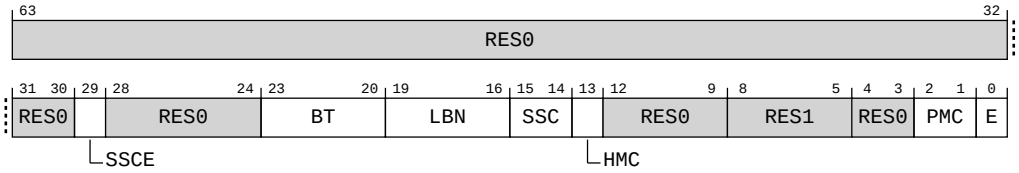


Table A-237: DBGBCR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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

Bits	Name	Description	Reset
[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, SSC, and SSCE. 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, SSC, and SSCE. 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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.15 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-91: AArch64_dbgwvr3_el1 bit assignments

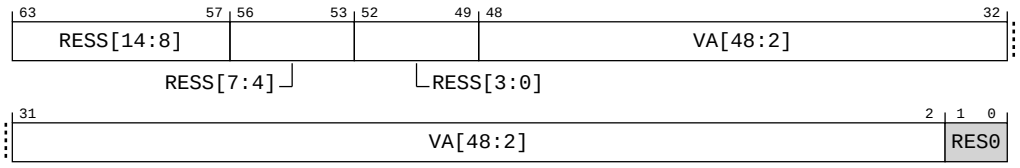


Table A-240: DBGWVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.16 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
```

```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-92: AArch64_dbgwcr3_el1 bit assignments

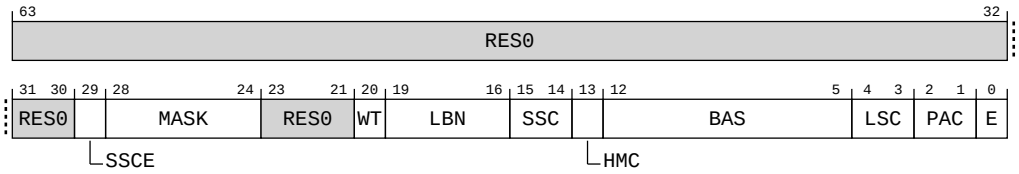


Table A-243: DBGWCR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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, SSC, and SSCE. 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-244: BAS description on page 371 In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-245: BAS description table 3 on page 371 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 nonzero 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.	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, SSC, and SSCE. 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. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table A-244: BAS description

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-245: BAS description table 3

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGWCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.17 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-93: AArch64_dbgbvr4_el1 bit assignments

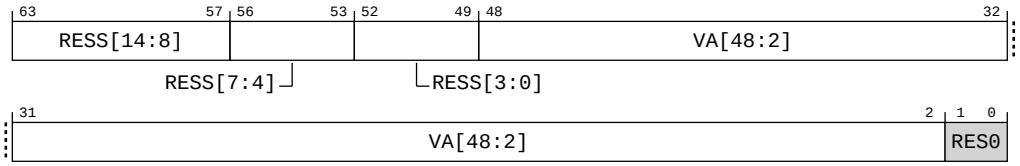


Table A-248: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-94: AArch64_dbgbvr4_el1 bit assignments

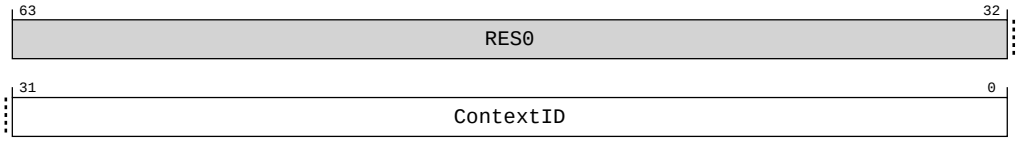


Table A-249: DBGBVR4_EL1 bit descriptions

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-95: AArch64_dbgbvr4_el1 bit assignments

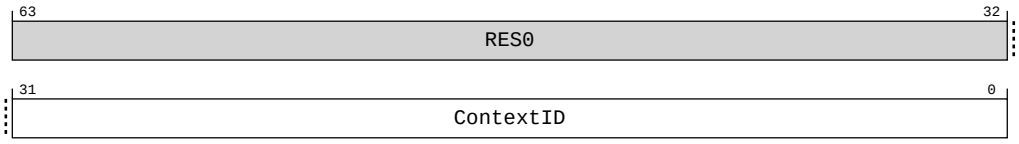


Table A-250: DBGBVR4_EL1 bit descriptions

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-96: AArch64_dbgbvr4_el1 bit assignments

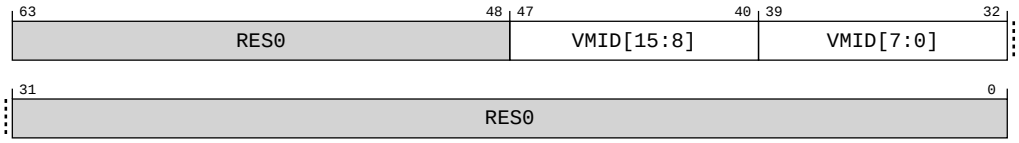


Table A-251: DBGBVR4_EL1 bit descriptions

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-97: AArch64_dbgbvr4_el1 bit assignments

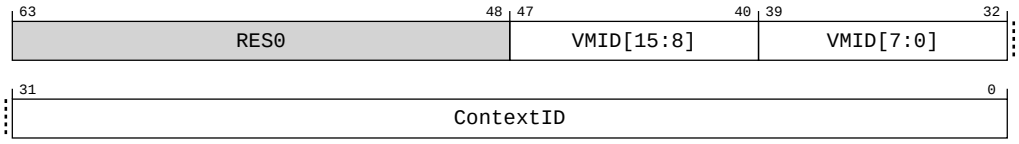


Table A-252: DBGBVR4_EL1 bit descriptions

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-98: AArch64_dbgbvr4_el1 bit assignments

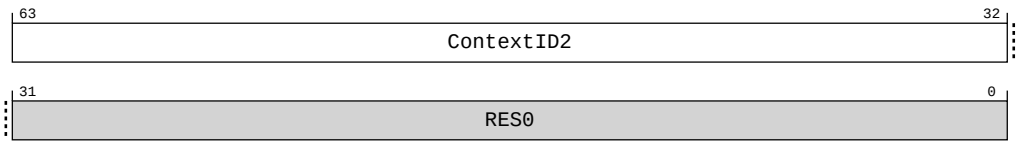


Table A-253: DBGBVR4_EL1 bit descriptions

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-99: AArch64_dbgbvr4_el1 bit assignments

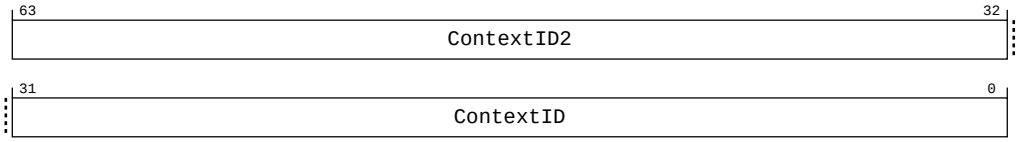


Table A-254: DBGBVR4_EL1 bit descriptions

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.18 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-100: AArch64_dbgbcr4_el1 bit assignments

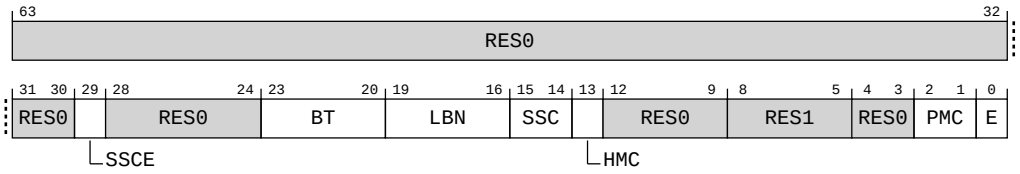


Table A-257: DBGBCR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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, SSC, and SSCE. 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, SSC, and SSCE. 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

Bits	Name	Description	Reset
[0]	E	Enable breakpoint n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.19 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-101: AArch64_dbgbvr5_el1 bit assignments

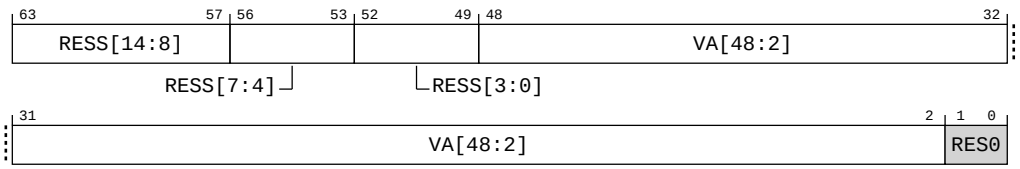


Table A-260: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-102: AArch64_dbgbvr5_el1 bit assignments

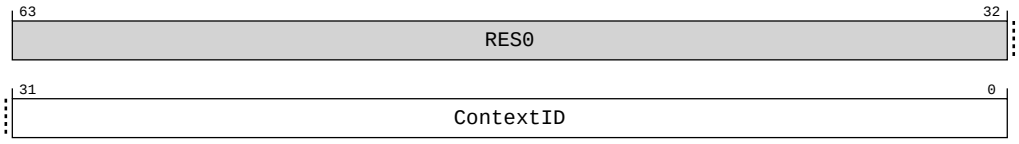


Table A-261: DBGBVR5_EL1 bit descriptions

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-103: AArch64_dbgbvr5_el1 bit assignments

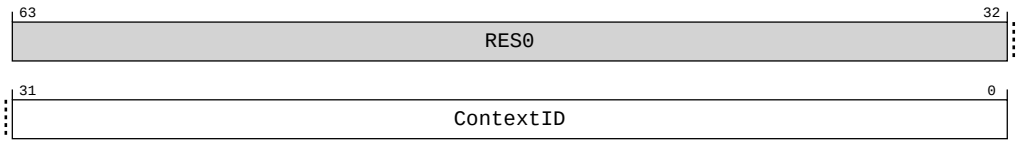


Table A-262: DBGBVR5_EL1 bit descriptions

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-104: AArch64_dbgbvr5_el1 bit assignments

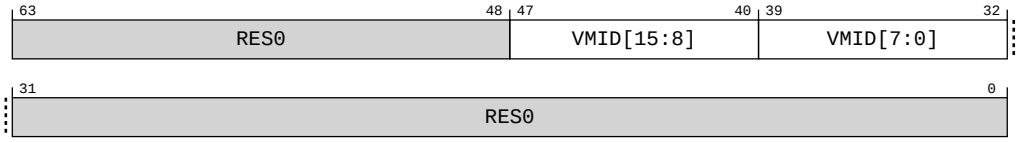


Table A-263: DBGBVR5_EL1 bit descriptions

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-105: AArch64_dbgbvr5_el1 bit assignments

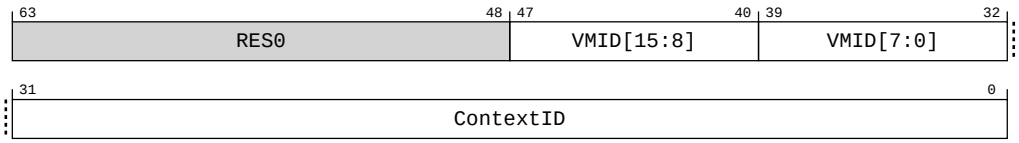


Table A-264: DBGBVR5_EL1 bit descriptions

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-DBGBCR5_EL1.BT == '110'

Figure A-106: AArch64_dbgbvr5_el1 bit assignments

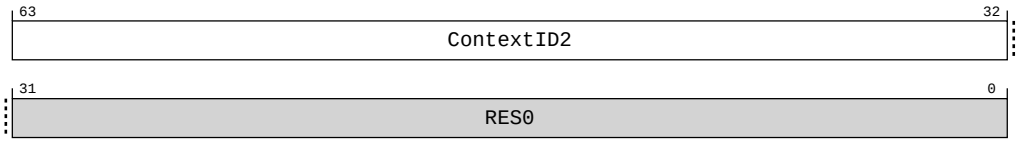


Table A-265: DBGBVR5_EL1 bit descriptions

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-107: AArch64_dbgbvr5_el1 bit assignments

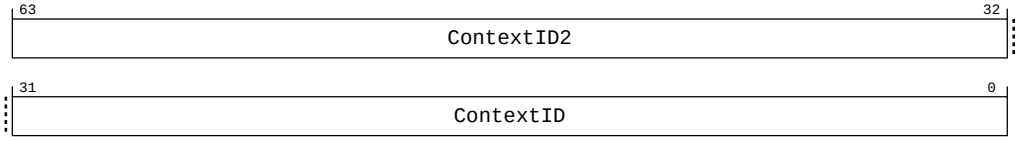


Table A-266: DBGBVR5_EL1 bit descriptions

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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBVRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.20 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-108: AArch64_dbgbcr5_el1 bit assignments

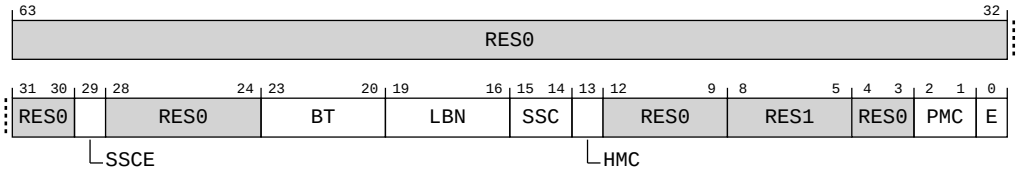


Table A-269: DBGBCR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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, SSC, and SSCE. 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

Bits	Name	Description	Reset
[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, SSC, and SSCE. 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, SSC, and SSCE. 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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.DBGBCRn_EL1 == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.DBGBCRn_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif 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.3.21 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-109: AArch64_imp_idata0_el3 bit assignments

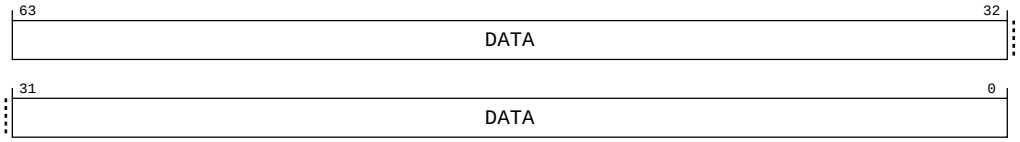


Table A-272: IMP_IDATA0_EL3 bit descriptions

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.3.22 IMP_IDATA1_EL3, Instruction Register 1

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-110: AArch64_imp_idata1_el3 bit assignments

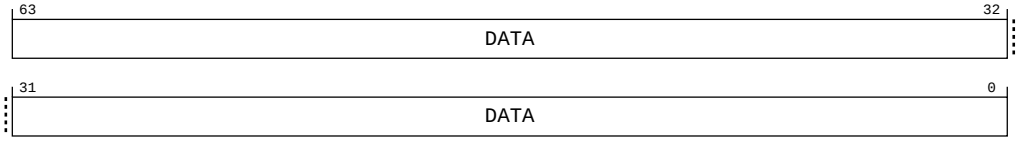


Table A-274: IMP_IDATA1_EL3 bit descriptions

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.3.23 IMP_IDATA2_EL3, Instruction Register 2

Contains data from a preceeding RAMINDEX operation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-111: AArch64_imp_idata2_el3 bit assignments

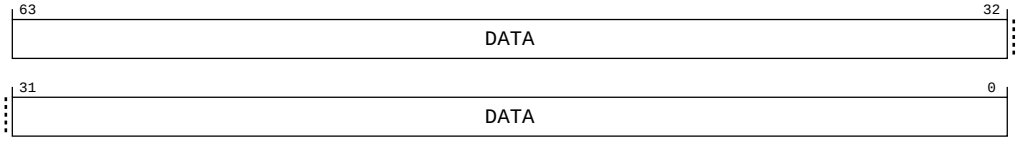


Table A-276: IMP_IDATA2_EL3 bit descriptions

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.3.24 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-112: AArch64_imp_ddata0_el3 bit assignments

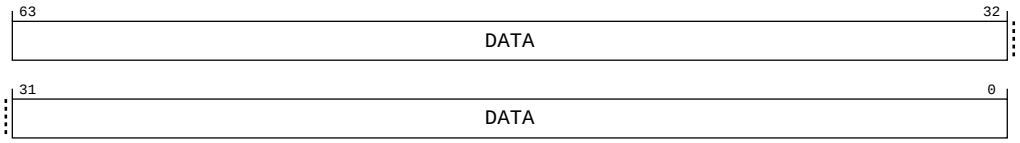


Table A-278: IMP_DDATA0_EL3 bit descriptions

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.3.25 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-113: AArch64_imp_ddata1_el3 bit assignments

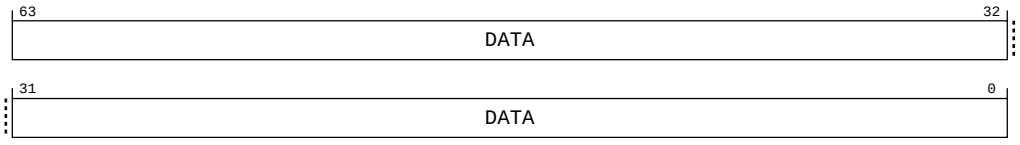


Table A-280: IMP_DDATA1_EL3 bit descriptions

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.3.26 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-114: AArch64_imp_ddata2_el3 bit assignments

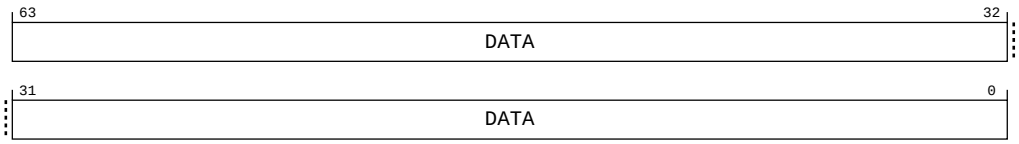


Table A-282: IMP_DDATA2_EL3 bit descriptions

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.4 AArch64 random number generator registers summary

The following summary table provides an overview of all AArch64 random number generator registers in the core.

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

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

Table A-284: registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPURNDBR_EL3	3	6	C15	C3	0	See individual bit resets.	64-bit	CPU Random Number Base Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPURNDPEID_EL3	3	6	C15	C3	1	See individual bit resets.	64-bit	CPU Random Number Packet Identification Register

A.4.1 IMP_CPURNDBR_EL3, CPU Random Number Base Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xx00 xxxx 0000 0000 0000 0000 0000 0000 0000 0000 xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-115: AArch64_imp_cpurndbr_el3 bit assignments

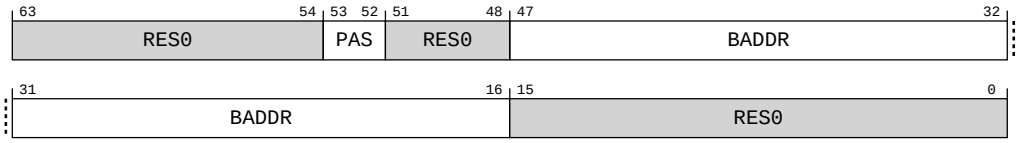


Table A-285: IMP_CPURNDBR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:54]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[53:52]	PAS	Indicates the physical address space of the external RNG block accesses. The possible values are: **0b00** Secure **0b01** Non-secure **0b10** Root (when LEGACY_TZ_EN is 0) **0b11** Realm (when LEGACY_TZ_EN is 0)	0b00
[51:48]	**RES0**	Reserved	**RES0**
[47:16]	BADDR	Indicates the base address bits [47:16] of the external RNG block	0x00000000
[15:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, S3_6_C15_C3_0

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0011	0b000

MSR S3_6_C15_C3_0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0011	0b000

Accessibility

MRS <Xt>, S3_6_C15_C3_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPURNDBR_EL3;
```

MSR S3_6_C15_C3_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPURNDBR_EL3 = X[t, 64];
```

A.4.2 IMP_CPURNDPEID_EL3, CPU Random Number Packet Identification Register

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000 0000 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-116: AArch64_imp_cpurndpeid_el3 bit assignments

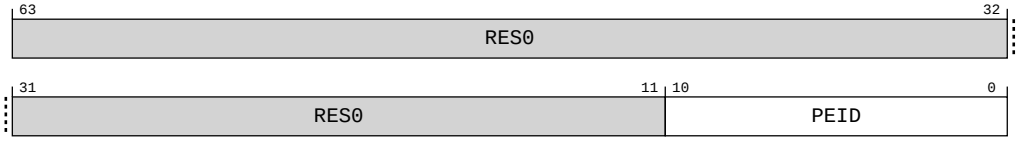


Table A-288: IMP_CPURNDPEID_EL3 bit descriptions

Bits	Name	Description	Reset
[63:11]	RES0	Reserved	RES0
[10:0]	PEID	Unique 11-bit hardware identification which is used to construct the address for RNDR accesses: RNDR address={CPURNDBR_EL3[47:16],CPURNDPEID_EL3[10:0],1'b0,4'b0}, RNDRRS address={CPURNDBR_EL3[47:16],CPURNDPEID_EL3[10:0],1'b1,4'b0}	0b00000000000

Access

MRS <Xt>, S3_6_C15_C3_1

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0011	0b001

MSR S3_6_C15_C3_1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b1111	0b0011	0b001

Accessibility

MRS <Xt>, S3_6_C15_C3_1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    X[t, 64] = IMP_CPURNDPEID_EL3;
```

MSR S3_6_C15_C3_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPURNDPEID_EL3 = X[t, 64];
```

A.5 AArch64 System instructions summary

The following summary table provides an overview of all System instructions in the core.

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

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

Table A-291: System instructions summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SYS_IMP_RAMINDEX	1	6	C15	C0	0	See individual bit resets.	64-bit	RAM Index

A.5.1 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-117: AArch64_sys_imp_ramindex bit assignments

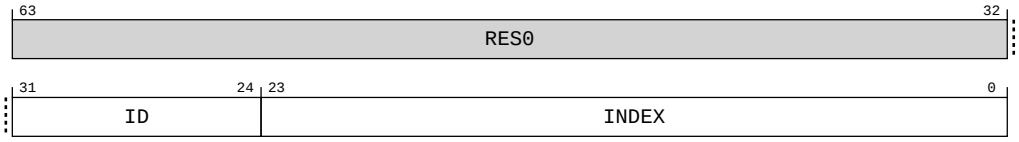


Table A-292: SYS_IMP_RAMINDEX bit descriptions

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.6 AArch64 Identification registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-294: Identification registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	0	C0	C0	0	See individual bit resets.	64-bit	Main ID Register
MPIDR_EL1	3	0	C0	C0	5	See individual bit resets.	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	0	C0	C0	6	See individual bit resets.	64-bit	Revision ID Register
ID_PFR0_EL1	3	0	C0	C1	0	See individual bit resets.	64-bit	AArch32 Processor Feature Register 0
ID_PFR1_EL1	3	0	C0	C1	1	See individual bit resets.	64-bit	AArch32 Processor Feature Register 1
ID_DFR0_EL1	3	0	C0	C1	2	See individual bit resets.	64-bit	AArch32 Debug Feature Register 0
ID_AFR0_EL1	3	0	C0	C1	3	See individual bit resets.	64-bit	AArch32 Auxiliary Feature Register 0
ID_MMFR0_EL1	3	0	C0	C1	4	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 0
ID_MMFR1_EL1	3	0	C0	C1	5	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 1
ID_MMFR2_EL1	3	0	C0	C1	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 2
ID_MMFR3_EL1	3	0	C0	C1	7	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 3
ID_ISAR0_EL1	3	0	C0	C2	0	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 0
ID_ISAR1_EL1	3	0	C0	C2	1	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 1
ID_ISAR2_EL1	3	0	C0	C2	2	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 2
ID_ISAR3_EL1	3	0	C0	C2	3	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	0	C0	C2	4	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 4
ID_ISAR5_EL1	3	0	C0	C2	5	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	0	C0	C2	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 4
ID_ISAR6_EL1	3	0	C0	C2	7	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	0	C0	C3	0	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	0	C0	C3	1	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	0	C0	C3	2	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	0	C0	C3	4	See individual bit resets.	64-bit	AArch32 Processor Feature Register 2

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ID_DFR1_EL1	3	0	C0	C3	5	See individual bit resets.	64-bit	Debug Feature Register 1
ID_MMFR5_EL1	3	0	C0	C3	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 5
ID_AA64PFR0_EL1	3	0	C0	C4	0	See individual bit resets.	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	0	C0	C4	1	See individual bit resets.	64-bit	AArch64 Processor Feature Register 1
ID_AA64ZFR0_EL1	3	0	C0	C4	4	See individual bit resets.	64-bit	SVE Feature ID register 0
ID_AA64DFR0_EL1	3	0	C0	C5	0	See individual bit resets.	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	0	C0	C5	1	See individual bit resets.	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	0	C0	C5	4	See individual bit resets.	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	0	C0	C5	5	See individual bit resets.	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	0	C0	C6	0	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	0	C0	C6	1	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64ISAR2_EL1	3	0	C0	C6	2	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 2
ID_AA64MMFR0_EL1	3	0	C0	C7	0	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	0	C0	C7	1	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	0	C0	C7	2	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 2
ID_AA64MMFR3_EL1	3	0	C0	C7	3	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 3
MPAMIDR_EL1	3	0	C10	C4	4	See individual bit resets.	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	0	C15	C0	0	See individual bit resets.	64-bit	CPU Configuration Register
CCSIDR_EL1	3	1	C0	C0	0	See individual bit resets.	64-bit	Current Cache Size ID Register
CLIDR_EL1	3	1	C0	C0	1	See individual bit resets.	64-bit	Cache Level ID Register
CCSIDR2_EL1	3	1	C0	C0	2	See individual bit resets.	64-bit	Current Cache Size ID Register 2
GMID_EL1	3	1	C0	C0	4	See individual bit resets.	64-bit	Multiple tag transfer ID register
CSSELR_EL1	3	2	C0	C0	0	See individual bit resets.	64-bit	Cache Size Selection Register
CTR_EL0	3	3	C0	C0	1	See individual bit resets.	64-bit	Cache Type Register
DCZID_EL0	3	3	C0	C0	7	See individual bit resets.	64-bit	Data Cache Zero ID register
VPIDR_EL2	3	4	C0	C0	0	See individual bit resets.	64-bit	Virtualization Processor ID Register
VMPIDR_EL2	3	4	C0	C0	5	See individual bit resets.	64-bit	Virtualization Multiprocessor ID Register

A.6.1 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 0000 1111 1101 1000 0100 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-118: AArch64_midr_el1 bit assignments

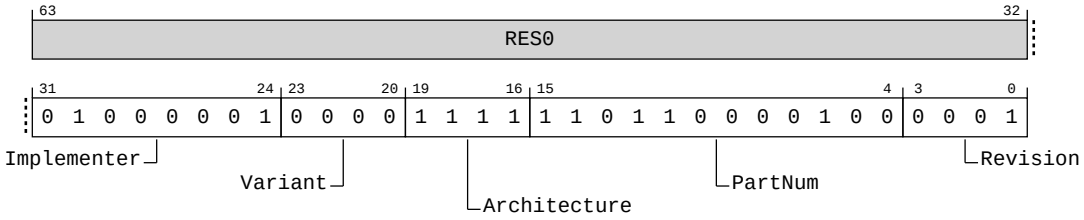


Table A-295: MIDR_EL1 bit descriptions

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. **0b0000** r0p1	0b0000
[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. **0b110110000100** Neoverse V3	0xD84

Bits	Name	Description	Reset
[3:0]	Revision	Revision number for the device. **0b0001** r0p1	0b0001

Access

MRS <Xt>, MIDR_EL1

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

Accessibility

MRS <Xt>, MIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.MIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() then
        X[t, 64] = VPIDR_EL2;
    else
        X[t, 64] = MIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MIDR_EL1;
```

A.6.2 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-119: AArch64_mpidr_el1 bit assignments

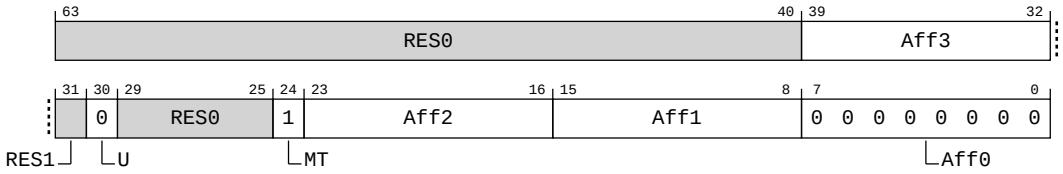


Table A-297: MPIDR_EL1 bit descriptions

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}

Bits	Name	Description	Reset
[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() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.MPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() then
        X[t, 64] = VMPIDR_EL2;
    else
        X[t, 64] = MPIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MPIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPIDR_EL1;
```

A.6.3 REVIDR_EL1, Revision ID Register

The REVIDR_EL1 provides revision information, additional to MIDR_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the Neoverse V3 core. Refer to the Neoverse V3 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-120: AArch64_revidr_el1 bit assignments

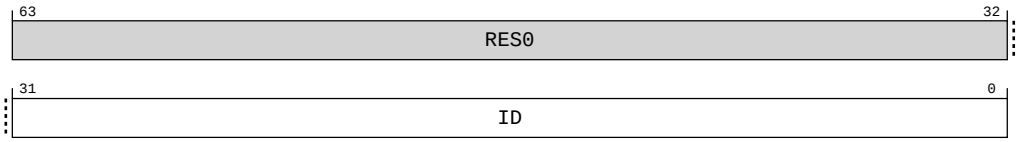


Table A-299: REVIDR_EL1 bit descriptions

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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.REVIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = REVIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = REVIDR_EL1;
```

```
elsif PSTATE.EL == EL3 then
    X[t, 64] = REVIDR_EL1;
```

A.6.4 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-121: AArch64_id_aa64pfr0_el1 bit assignments

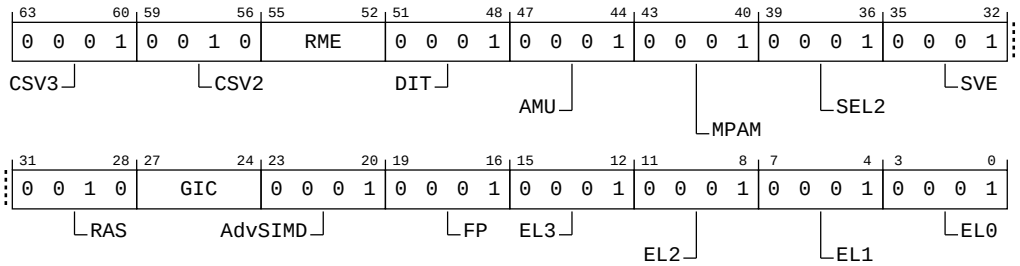


Table A-301: ID_AA64PFR0_EL1 bit descriptions

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]	RME	Realm Management Extension (RME). Defined values are: **0b0000** When Port LEGACY_TZ_EN is High, Realm Management Extension not implemented. **0b0001** When Port LEGACY_TZ_EN is Low, RMEv1 is implemented.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[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

Bits	Name	Description	Reset
[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
[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.6.5 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

```
0000 0000 0000 0000 0000 0000 0000 0000 0001 0000 xxxx 0001 xxxx xxxx 0010 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-122: AArch64_id_aa64pfr1_el1 bit assignments

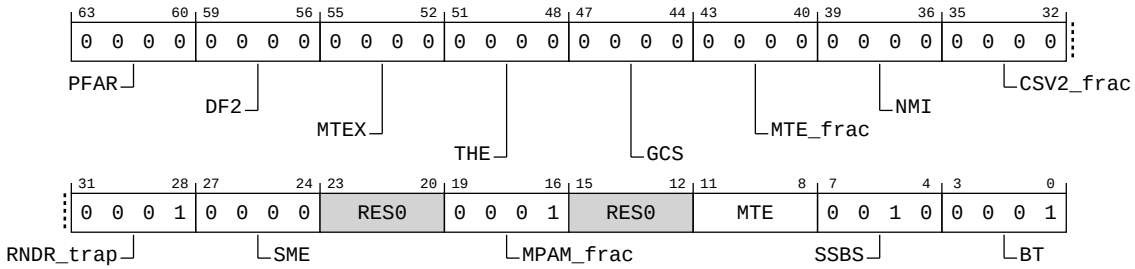


Table A-303: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	PFAR	Support for physical fault address registers, FEAT_PFAR. Defined values are: **0b0000** FEAT_PFAR is not implemented.	0b0000
[59:56]	DF2	Support for error exception routing extensions, FEAT_DoubleFault2. Defined values are: **0b0000** FEAT_DoubleFault2 is not implemented. **Note:** This does not mean that FEAT_DoubleFault, as identified by AArch64-ID_AA64PFR0_EL1.RAS >= 0b0010, is not implemented.	0b0000
[55:52]	MTEX	Additional tag checking modes for MTE. Defined values are: **0b0000** Support for Memory Tagging when Address tagging is enabled.	0b0000
[51:48]	THE	Support for Translation Hardening Extension. Defined values are: **0b0000** Translation Hardening Extension is not implemented.	0b0000
[47:44]	GCS	Support for Guarded Control Stack. Defined values are: **0b0000** Guarded Control Stack is not implemented.	0b0000
[43:40]	MTE_frac	Support for Asynchronous Faulting and asymmetric Tag Check Fault handling. Defined values are: **0b0000** Asynchronous Faulting is supported. If ID_AA64PFR1_EL1.MTE >= 0b0011, asymmetric Tag Check Fault handling is supported.	0b0000
[39:36]	NMI	Non-maskable Interrupt. Indicates support for Non-maskable interrupts. Defined values are: **0b0000** SCTLR_ELx.{SPINTMASK, NMI} and PSTATE.ALLINT with its associated instructions are not supported.	0b0000
[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: **0b0001** Trapping of AArch64-RNDR and AArch64-RNDRRS to EL3 is supported. AArch64-SCR_EL3.TRNDR is present.	0b0001

Bits	Name	Description	Reset
[27:24]	SME	Scalable Matrix Extension. Defined values are: **0b0000** SME architectural state and programmers' model are not implemented.	0b0000
[23:20]	**RES0**	Reserved	**RES0**
[19:16]	MPAM_frac	Indicates the minor version number of support for the MPAM Extension. Defined values are: **0b0001** The minor version number of the MPAM extension is 1.	0b0001
[15:12]	**RES0**	Reserved	**RES0**
[11:8]	MTE	Support for the Memory Tagging Extension. 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.6.6 ID_AA64ZFR0_EL1, SVE Feature ID register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension instruction set, when one or more of FEAT_SVE and FEAT_SME is implemented.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

If FEAT_SME is implemented and FEAT_SVE is not implemented, then SVE instructions can only be executed when the PE is in Streaming SVE mode and the instructions are legal to execute in Streaming SVE mode.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
xxxx 0000 0000 xxxx 0001 xxxx xxxx xxxx xxxx 0000 0001 0001 xxxx xxxx xxxx 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-123: AArch64_id_aa64zfr0_el1 bit assignments

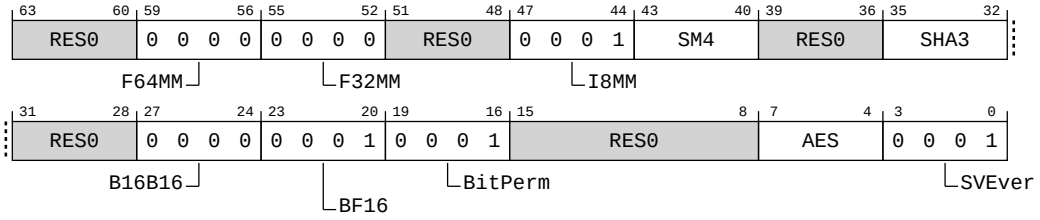


Table A-305: ID_AA64ZFR0_EL1 bit descriptions

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.	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.	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.
[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.
[31:28]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[27:24]	B16B16	Indicates support for SVE2.1 non-widening BFloat16 instructions. Defined values are: **0b0000** SVE2.1 non-widening BFloat16 instructions are not implemented.	0b0000
[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 one or more of FEAT_SME and FEAT_SVE is implemented. Defined values are: **0b0000** The SVE instructions are implemented. **0b0001** As 0b0000, and adds the mandatory SVE2 instructions. `For this product, the selected value is 0b0001.`	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.6.7 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

The external register ext-EDDFR gives information from this register.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0001 0000 0001 1111 0001 0001 1111 0011 0001 0000 0011 0000 0101 0111 0001 1001

Bit descriptions

Figure A-124: AArch64_id_aa64dfr0_el1 bit assignments

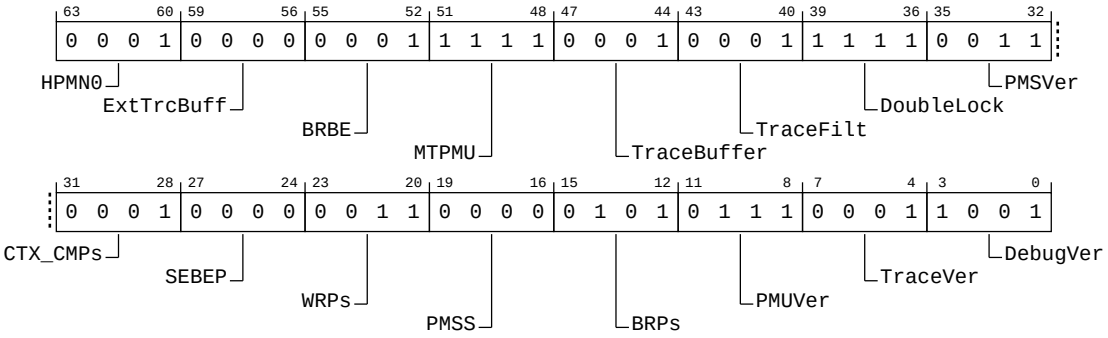


Table A-307: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	HPMN0	Zero PMU event counters for a Guest operating system. Defined values are: **0b0001** Setting AArch64-MDCR_EL2.HPMN to zero has defined behavior.	0b0001
[59:56]	ExtTrcBuff	Trace Buffer External Mode Extension. Defined values are: **0b0000** Trace Buffer External Mode not implemented.	0b0000
[55:52]	BRBE	Branch Record Buffer Extension. Defined values are: **0b0001** Branch Record Buffer Extension implemented.	0b0001
[51:48]	MTPMU	Multi-threaded PMU extension. Defined values are: **0b1111** FEAT_MTPMU not implemented. If FEAT_PMUv3 is implemented, AArch64-PMEVTYPER<n>_EL0.MT and AArch32-PMEVTYPER<n>.MT are RES0.	0b1111
[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: **0b0011** As 0b0010, and adds: • Discard mode. • Extended event filtering, including the AArch64-PMSNEVFR_EL1 System register. • Support for the **OPTIONAL** previous branch target Address packet. • If FEAT_PMUv3 is implemented, controls to freeze the PMU event counters after an SPE buffer management event occurs. • If FEAT_PMUv3 is implemented, the SAMPLE_FEED_BR, SAMPLE_FEED_EVENT, SAMPLE_FEED_LAT, SAMPLE_FEED_LD, SAMPLE_FEED_OP, and SAMPLE_FEED_ST PMU events.	0b0011
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. **0b0001** Two context-aware breakpoints are included	0b0001
[27:24]	SEBEP	Synchronous-exception-based event profiling. Defined values are: **0b0000** Synchronous-exception-based event profiling not implemented.	0b0000
[23:20]	WRPs	Number of watchpoints, minus 1. **0b0011** Four Watchpoints	0b0011

Bits	Name	Description	Reset
[19:16]	PMSS	PMU Snapshot extension. Defined values are: **0b0000** PMU snapshot extension not implemented.	0b0000
[15:12]	BRPs	Number of breakpoints, minus 1. **0b0101** Six Breakpoints	0b0101
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **0b0111** Performance Monitors Extension implemented, PMUv3 for Armv8.7	0b0111
[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.6.8 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1

Provides top level information about the debug system in AArch64.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx 0000 0000 xxxx xxxx xxxx xxxx 0000 0000 0000 0000 0000 0000 xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-125: AArch64_id_aa64dfr1_el1 bit assignments

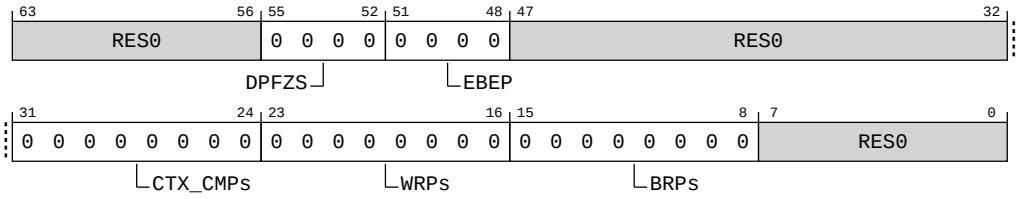


Table A-309: ID_AA64DFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:52]	DPFZS	Behavior of the cycle counter when event counting is frozen by a Statistical Profiling management event. Defined values are: **0b0000** The cycle counter AArch64-PMCCNTR_EL0 is never affected by AArch64-PMCR_EL0.FZS.	0b0000
[51:48]	EBEP	Exception-based event profiling. Defined values are: **0b0000** Exception-based event profiling not implemented.	0b0000
[47:32]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[31:24]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. The value `0x00` means that the number of breakpoints that are context-aware is described by AArch64-ID_AA64DFR0_EL1.CTX_CMPs. Otherwise, the value of this field is the number of breakpoints that are context-aware, minus 1. Defined values are: **0b00000000** AArch64-ID_AA64DFR0_EL1.CTX_CMPs is the number of breakpoints that are context-aware, minus 1.	`0x00`
[23:16]	WRPs	Number of watchpoints, minus 1. The value `0x00` means that the number of watchpoints is described by AArch64-ID_AA64DFR0_EL1.WRPs. Otherwise, the value of this field is the number of watchpoints, minus 1. Defined values are: **0b00000000** AArch64-ID_AA64DFR0_EL1.WRPs is the number of watchpoints, minus 1.	`0x00`
[15:8]	BRPs	Number of breakpoints, minus 1. The value `0x00` means that the number of breakpoints is described by AArch64-ID_AA64DFR0_EL1.BRPs. Otherwise, the value of this field is the number of breakpoints, minus 1. Defined values are: **0b00000000** AArch64-ID_AA64DFR0_EL1.BRPs is the number of breakpoints, minus 1.	`0x00`
[7:0]	**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.6.9 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-126: AArch64_id_aa64afr0_el1 bit assignments

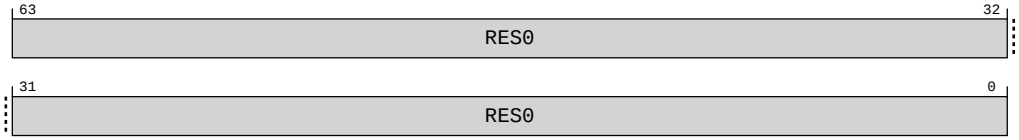


Table A-311: ID_AA64AFR0_EL1 bit descriptions

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.6.10 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

Reserved for future expansion of information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-127: AArch64_id_aa64afr1_el1 bit assignments

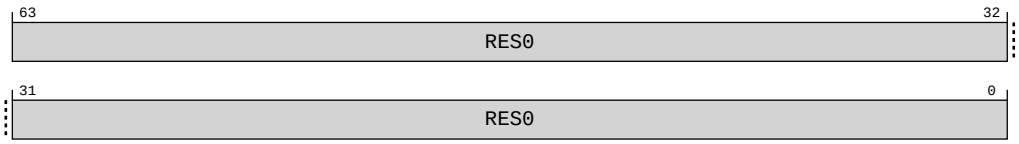


Table A-313: ID_AA64AFR1_EL1 bit descriptions

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.6.11 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
0001 0010 0010 0001 0001 xxxx xxxx xxxx 0001 0000 0010 0001 xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-128: AArch64_id_aa64isar0_el1 bit assignments

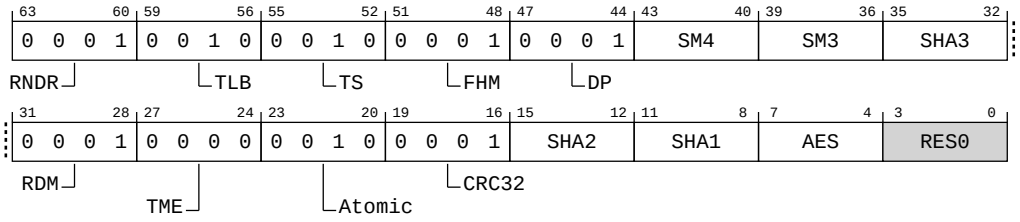


Table A-315: ID_AA64ISAR0_EL1 bit descriptions

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: **0b0001** AArch64-RNDR and AArch64-RNDRRS registers are implemented.	0b0001
[59:56]	TLB	Indicates support for Outer Shareable and TLB range maintenance instructions. Defined values are: **0b0010** Outer Shareable and TLB range maintenance instructions are implemented.	0b0010

Bits	Name	Description	Reset
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	0b0010
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: **0b0001** FMLAL and FMLSL instructions are implemented.	0b0001
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **0b0001** UDOT and SDOT instructions implemented.	0b0001
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **0b0000** 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

Bits	Name	Description	Reset
[27:24]	TME	Indicates support for TME instructions. Defined values are: **0b0000** TME instructions are not implemented. **When PSTATE.EL IN {EL2, EL1}** Access to this field is: **RAZ/WI** **When PSTATE.EL == EL1 && EL2Enabled()** Access to this field is: **RAZ/WI** **Otherwise** Access to this field is: RO	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 are implemented.	0b0001
[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.

Bits	Name	Description	Reset
[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.6.12 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0011 0001 0001 0001 0001 0001 0001 0001 0000 0000 0010 0001 0001 0000 0000 0010

Bit descriptions

Figure A-129: AArch64_id_aa64isar1_el1 bit assignments

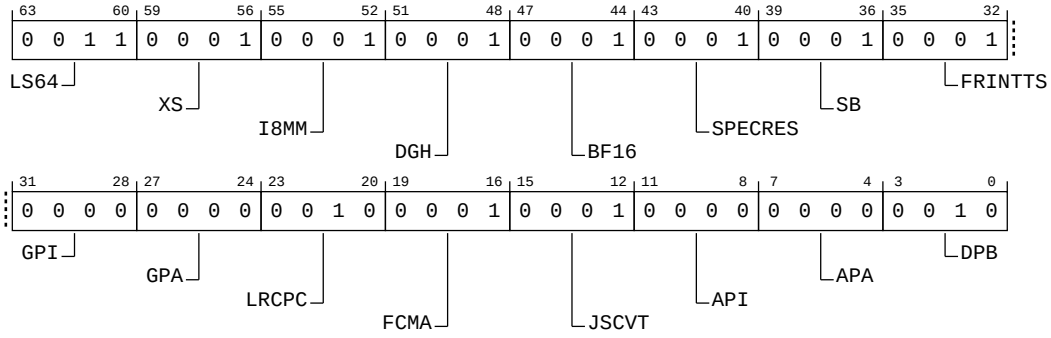


Table A-317: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	LS64	Indicates support for LD64B and ST64B* instructions, and the AArch64-ACCDATA_EL1 register. Defined values of this field are: **0b0011** The LD64B, ST64B, ST64BV, and ST64BV0 instructions, the AArch64-ACCDATA_EL1 register, and their associated traps are supported.	0b0011
[59:56]	XS	Indicates support for the XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the AArch64-HCRX_EL2.{FGTnXS, FnXS} fields in AArch64 state. Defined values are: **0b0001** The XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the AArch64-HCRX_EL2.{FGTnXS, FnXS} fields are supported.	0b0001
[55:52]	I8MM	Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values are: **0b0001** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	0b0001
[51:48]	DGH	Indicates support for the Data Gathering Hint instruction. Defined values are: **0b0001** Data Gathering Hint is implemented.	0b0001

Bits	Name	Description	Reset
[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: **0b0000** Generic Authentication using the QARMA5 algorithm is not implemented.	0b0000
[23:20]	LRCPC	Indicates support for weaker release consistency, RCpc, based model. Defined values are: **0b0010** As 0b0001, and the LDAPR (unscaled immediate) and STLR (unscaled immediate) 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: **0b0000** Address Authentication using the QARMA5 algorithm is not implemented.	0b0000

Bits	Name	Description	Reset
[3:0]	DPB	Data Persistence writeback. Indicates support for the DC CVAP and DC CVADP instructions in AArch64 state. Defined values are: **0b0010** DC CVAP and DC CVADP supported.	0b0010

Access

MRS <Xt>, ID_AA64ISAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b001

Accessibility

MRS <Xt>, ID_AA64ISAR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        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.6.13 ID_AA64ISAR2_EL1, AArch64 Instruction Set Attribute Register 2

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations


Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx 0000 0000 xxxx 0000 0000 0000 0000 0001 0000 0000 0101 0001 0000 0010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-130: AArch64_id_aa64isar2_el1 bit assignments

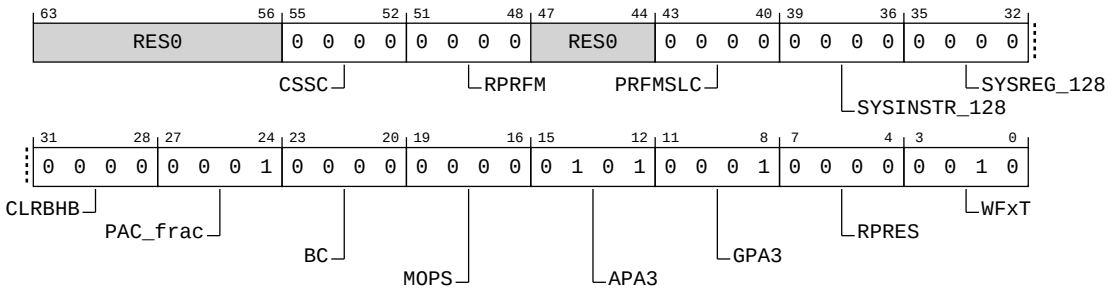


Table A-319: ID_AA64ISAR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	**RES0**	Reserved	**RES0**
[55:52]	CSSC	Indicates support for common short sequence compression instructions. Defined values are: **0b0000** Common short sequence compression instructions are not implemented.	0b0000
[51:48]	RPRFM	RPRFM hint instruction. Defined values are: **0b0000** RPRFM hint instruction is not implemented and is treated as a **NOP**.	0b0000
[47:44]	**RES0**	Reserved	**RES0**
[43:40]	PRFMSLC	Indicates whether the PRFM instructions support a system level cache option. Defined values are: **0b0000** The PRFM instructions do not support the SLC target.	0b0000

Bits	Name	Description	Reset
[39:36]	SYSINSTR_128	SYSINSTR_128. Indicates support for System instructions that can take 128-bit inputs. Defined values are: **0b0000** System instructions that can take 128-bit inputs are not supported.	0b0000
[35:32]	SYSREG_128	SYSREG_128. Indicates support for instructions to access 128-bit System Registers. Defined values are: **0b0000** Instructions to access 128-bit System Registers are not supported.	0b0000
[31:28]	CLRBHB	Indicates support for the CLRBHB instruction in AArch64 state. Defined values are: **0b0000** CLRBHB instruction is not implemented.	0b0000
[27:24]	PAC_frac	Indicates whether the ConstPACField() function used as part of the PAC addition returns FALSE or TRUE. **0b0001** ConstPACField() returns TRUE.	0b0001
[23:20]	BC	Indicates support for the BC instruction in AArch64 state. Defined values are: **0b0000** BC instruction is not implemented.	0b0000
[19:16]	MOPS	Indicates support for the Memory Copy and Memory Set instructions in AArch64 state. **0b0000** The Memory Copy and Memory Set instructions are not implemented in AArch64 state.	0b0000
[15:12]	APA3	Indicates whether the QARMA3 algorithm is implemented in the PE for address authentication in AArch64 state. This applies to all Pointer Authentication instructions other than the `PACGA` instruction. Defined values are: **0b0101** Address Authentication using the QARMA3 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.	0b0101
[11:8]	GPA3	Indicates whether the QARMA3 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0b0001** Generic Authentication using the QARMA3 algorithm is implemented. This includes the `PACGA` instruction.	0b0001
[7:4]	RPRES	Indicates support for 12 bits of mantissa in reciprocal and reciprocal square root instructions in AArch64 state, when AArch64-FPCR.AH is 1. Defined values are: **0b0000** Reciprocal and reciprocal square root estimates give 8 bits of mantissa, when AArch64-FPCR.AH is 1.	0b0000
[3:0]	WFxT	Indicates support for the `WFET` and `WFIT` instructions in AArch64 state. Defined values are: **0b0010** `WFET` and `WFIT` are supported, and the register number is reported in the ESR_ELx on exceptions.	0b0010

Access

MRS <Xt>, ID_AA64ISAR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b010

Accessibility

MRS <Xt>, ID_AA64ISAR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR2_EL1;
```

A.6.14 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
0010 0001 xxxx xxxx 0000 0010 0010 0010 0000 0000 0001 xxxx 0001 0001 0010 0101
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-131: AArch64_id_aa64mmfr0_el1 bit assignments

Table A-321: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	ECV	Indicates presence of Enhanced Counter Virtualization. Defined values are: **0b0010** As 0b0001, and also includes support for AArch64-CNTHCTL_EL2.ECV and AArch64-CNTPOFF_EL2.	0b0010
[59:56]	FGT	Indicates presence of the Fine-Grained Trap controls. Defined values are: **0b0001** Fine-grained trap controls are implemented. Supports: • If EL2 is implemented, the AArch64-HAFGRTR_EL2, AArch64-HDFGRTR_EL2, AArch64-HDFGWTR_EL2, AArch64-HFGRTR_EL2, AArch64-HFGITR_EL2 and AArch64-HFGWTR_EL2 registers, and their associated traps. • If EL2 is implemented, AArch64-MDCR_EL2.TDCC. • If EL3 is implemented, AArch64-MDCR_EL3.TDCC. • If both EL2 and EL3 are implemented, AArch64-SCR_EL3.FGTEn.	0b0001
[55:48]	**RES0**	Reserved	**RES0**
[47:44]	ExS	Indicates support for disabling context synchronizing exception entry and exit. Defined values are: **0b0000** All exception entries and exits are context synchronization events.	0b0000
[43:40]	TGran4_2	Indicates support for 4KB memory granule size at stage 2. Defined values are: **0b0010** 4KB granule supported at stage 2.	0b0010
[39:36]	TGran64_2	Indicates support for 64KB memory granule size at stage 2. Defined values are: **0b0010** 64KB granule supported at stage 2.	0b0010

Bits	Name	Description	Reset
[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: **0b0101** 48 bits, 256TB.	0b0101

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.6.15 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

0001 0000 0000 0001 0001 0001 0001 0000 0001 0000 0011 0001 0010 0001 0010 0010

Bit descriptions

Figure A-132: AArch64_id_aa64mmfr1_el1 bit assignments

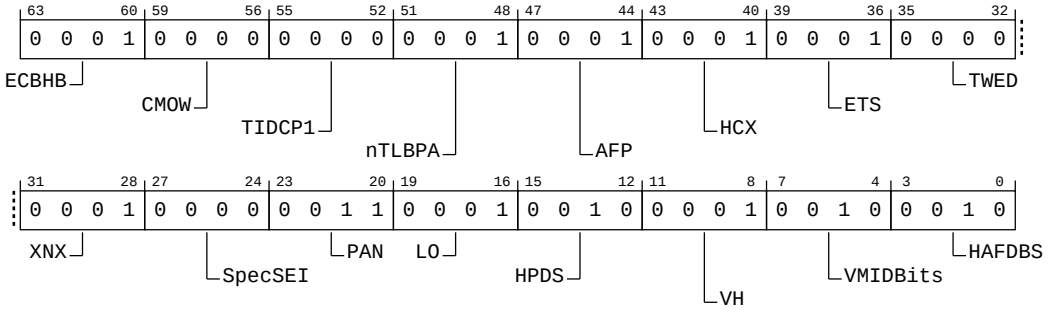


Table A-323: ID_AA64MMFR1_EL1 bit descriptions

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
[59:56]	CMOW	Indicates support for cache maintenance instruction permission. Defined values are: **0b0000** AArch64-SCTLR_EL1.CMOW, AArch64-SCTLR_EL2.CMOW, and AArch64-HCRX_EL2.CMOW bits are not implemented.	0b0000
[55:52]	TIDCP1	Indicates whether AArch64-SCTLR_EL1.TIDCP and AArch64-SCTLR_EL2.TIDCP are implemented in AArch64 state. Defined values are: **0b0000** AArch64-SCTLR_EL1.TIDCP and AArch64-SCTLR_EL2.TIDCP bits are not implemented and are **RES0**.	0b0000
[51:48]	nTLBPA	Indicates support for intermediate caching of translation table walks. Defined values are: **0b0001** The intermediate caching of translation table walks does not include non-coherent physical translation caches.	0b0001
[47:44]	AFP	Indicates support for AArch64-FPCR.{AH, FIZ, NEP}. Defined values are: **0b0001** The AArch64-FPCR.{AH, FIZ, NEP} fields are supported.	0b0001
[43:40]	HCX	Indicates support for AArch64-HCRX_EL2 and its associated EL3 trap. Defined values are: **0b0001** AArch64-HCRX_EL2 and its associated EL3 trap are supported.	0b0001
[39:36]	ETS	Indicates support for Enhanced Translation Synchronization. Defined values are: **0b0001** Enhanced Translation Synchronization is supported.	0b0001
[35:32]	TWED	Indicates support for the configurable delayed trapping of WFE. Defined values are: **0b0000** Configurable delayed trapping of WFE is not supported.	0b0000
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. 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

Bits	Name	Description	Reset
[19:16]	LO	LORegions. Indicates support for LORegions. Defined values are: **0b0001** LORegions supported.	0b0001
[15:12]	HPDS	Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are: **0b0010** Disabling of hierarchical controls supported with the TCR_EL1.{HPD1, HPD0}, TCR_EL2.HPD or TCR_EL2.{HPD1, HPD0}, and TCR_EL3.HPD bits and adds possible hardware allocation of bits[62:59] of the translation table descriptors from the final lookup level for **IMPLEMENTATION DEFINED** use.	0b0010
[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** As 0b0001, and adds support for hardware update of the Access flag for Block and Page descriptors. Hardware update of 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.6.16 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
0001 0010 0010 0001 xxxx 0001 0001 0001 0001 0010 0001 0000 0001 0000 0001 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-133: AArch64_id_aa64mmfr2_el1 bit assignments

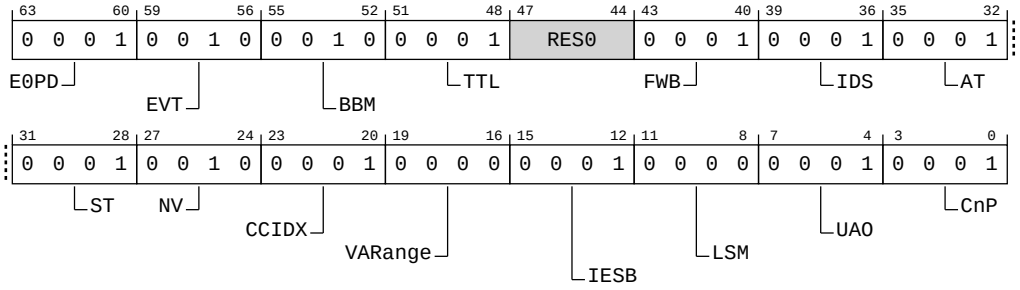


Table A-325: ID_AA64MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	E0PD	Indicates support for the E0PD mechanism. Defined values are: **0b0001** E0PDx mechanism is implemented.	0b0001
[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: **0b0010** The AArch64-VNCR_EL2 register and the AArch64-HCR_EL2.{NV2, AT, NV1, NV} bits are implemented.	0b0010
[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

Bits	Name	Description	Reset
[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
[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.6.17 ID_AA64MMFR3_EL1, AArch64 Memory Model Feature Register 3

Provides information about the implemented memory model and memory management support in AArch64 state.

Configurations

Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

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-134: AArch64_id_aa64mmfr3_el1 bit assignments

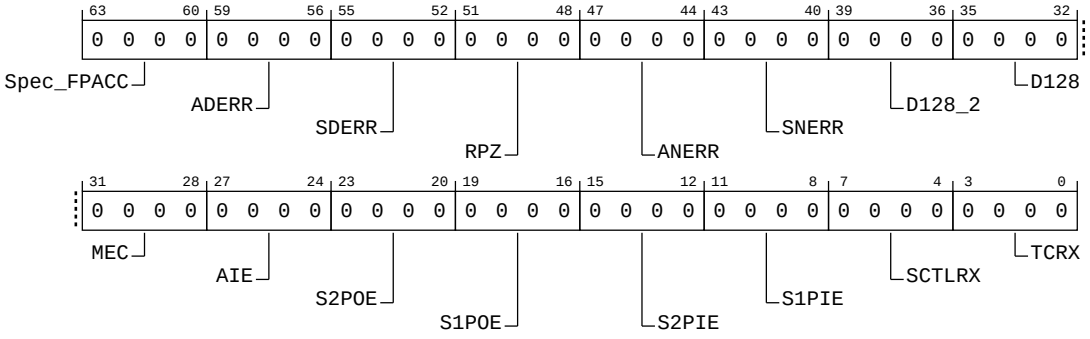


Table A-327: ID_AA64MMFR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	Spec_FPACC	Speculative behavior in the event of a PAC authentication failure in an implementation that includes FEAT_FPACCOMBINE. Defined values are: **0b0000** The implementation does not disclose whether the speculative use of pointers processed by a PAC Authentication is materially different in terms of the impact on cached microarchitectural state between passing and failing of the PAC Authentication.	0b0000
[59:56]	ADERR	Asynchronous Device error exceptions. With ID_AA64MMFR3_EL1.SDERR, describes the PE behavior for error exceptions on Device memory loads. Defined values are: **0b0000** If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.SDERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.SDERR.	0b0000
[55:52]	SDERR	Synchronous Device error exceptions. With ID_AA64MMFR3_EL1.ADERR, describes the PE behavior for error exceptions on Device memory loads. Defined values are: **0b0000** If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.ADERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.ADERR.	0b0000
[51:48]	RPZ	Remove Poison and Zero. Describes support for the Remove Poison and Zero instruction, `DC RPGDZPA`. Defined values are: **0b0000** FEAT_RPZ is not implemented.	0b0000
[47:44]	ANERR	Asynchronous Normal error exceptions. With ID_AA64MMFR3_EL1.SNERR, describes the PE behavior for error exceptions on Normal memory loads. Defined values are: **0b0000** If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.SNERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.SNERR.	0b0000
[43:40]	SNERR	Synchronous Normal error exceptions. With ID_AA64MMFR3_EL1.ANERR, describes the PE behavior for error exceptions on Normal memory loads. Defined values are: **0b0000** If FEAT_RASv2 is not implemented and ID_AA64MMFR3_EL1.ANERR is 0b0000, then the behavior is not described. Otherwise, the behavior is described by ID_AA64MMFR3_EL1.ANERR.	0b0000
[39:36]	D128_2	128-bit Page Table Descriptor at stage 2. Indicates support for 128-bit Page Table Descriptor at stage 2. Defined values are: **0b0000** 128-bit Page Table Descriptor Extension at stage 2 is not supported.	0b0000
[35:32]	D128	128-bit Page Table Descriptor. Indicates support for 128-bit Page Table Descriptor. Defined values are: **0b0000** 128-bit Page Table Descriptor Extension is not supported.	0b0000
[31:28]	MEC	Indicates support for Memory Encryption Contexts. Defined values are: **0b0000** Memory Encryption Contexts is not supported.	0b0000
[27:24]	AIE	Attribute Indexing. Indicates support for the Attribute Index Enhancement. Defined values are: **0b0000** The Attribute Index Enhancement is not supported.	0b0000

Bits	Name	Description	Reset
[23:20]	S2POE	Stage 2 Permission Overlay. Indicates support for Permission Overlay at Stage 2. Defined values are: **0b0000** Permission Overlay at Stage 2 is not supported.	0b0000
[19:16]	S1POE	Stage 1 Permission Overlay. Indicates support for Permission Overlay at Stage 1. Defined values are: **0b0000** Permission Overlay at Stage 1 is not supported.	0b0000
[15:12]	S2PIE	Stage 2 Permission Indirection. Indicates support for Permission Indirection at Stage 2. Defined values are: **0b0000** Permission Indirection at Stage 2 is not supported.	0b0000
[11:8]	S1PIE	Stage 1 Permission Indirection. Indicates support for Permission Indirection at Stage 1. Defined values are: **0b0000** Permission Indirection at Stage 1 is not supported.	0b0000
[7:4]	SCTLRX	SCTLRX Extension. Indicates support for Extension of AArch64-SCTLR_EL1. Defined values are: **0b0000** AArch64-SCTLR2_EL1, AArch64-SCTLR2_EL2 and their associated trap controls are not implemented.	0b0000
[3:0]	TCRX	TCR Extension. Indicates support for Extension of AArch64-TCR_EL1. Defined values are: **0b0000** AArch64-TCR2_EL1, AArch64-TCR2_EL2 and their associated trap controls are not implemented.	0b0000

Access

MRS <Xt>, ID_AA64MMFR3_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b011

Accessibility

MRS <Xt>, ID_AA64MMFR3_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR3_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR3_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64MMFR3_EL1;
```

A.6.18 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

```
xx10 111x xxxx xxxx xxxx xxxx 0000 0001 xxxx xxxx xxx1 111x 0000 0111 1111 1111
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

MPAMIDR_EL1 indicates the MPAM implementation parameters of the PE.

Figure A-135: AArch64_mpamidr_el1 bit assignments

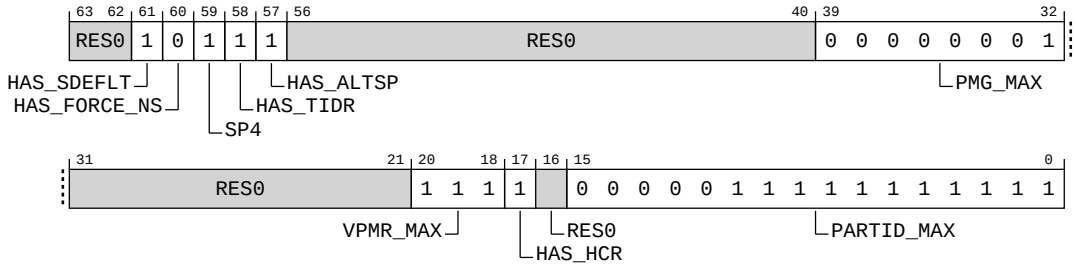


Table A-329: MPAMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:62]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[61]	HAS_SDEFLT	HAS_SDEFLT indicates support for AArch64-MPAM3_EL3.SDEFLT bit. Defined values are: **0b1** The SDEFLT bit is implemented in AArch64-MPAM3_EL3.	0b1
[60]	HAS_FORCE_NS	HAS_FORCE_NS indicates support for AArch64-MPAM3_EL3.FORCE_NS bit. Defined values are: **0b0** The FORCE_NS bit is not implemented in AArch64-MPAM3_EL3.	0b0
[59]	SP4	Supports 4 MPAM PARTID spaces. **0b1** MPAM supports 4 PARTID spaces.	0b1
[58]	HAS_TIDR	HAS_TIDR indicates support for AArch64-MPAM2_EL2.TIDR bit. Defined values are: **0b1** The TIDR bit is implemented in AArch64-MPAM2_EL2.	0b1
[57]	HAS_ALTSP	HAS_ALTSP indicates support for alternative PARTID spaces. **0b1** Alternative PARTID spaces are implemented with control bits in AArch64-MPAM3_EL3 and AArch64-MPAM2_EL2.	0b1
[56:40]	**RES0**	Reserved	**RES0**
[39:32]	PMG_MAX	The largest value of PMG that the implementation can generate. The PMG_I and PMG_D fields of every MPAMn_ELx must implement at least enough bits to represent PMG_MAX. **0b00000001** Max PMG field is 1 (1-bit)	0x01
[31:21]	**RES0**	Reserved	**RES0**
[20:18]	VPMR_MAX	Indicates the maximum register index n for the MPAMVPM<n>_EL2 registers. **0b111** 8 MPAMVPMn_EL2 registers are implemented	0b111
[17]	HAS_HCR	HAS_HCR indicates that the PE implementation supports MPAM virtualization, including AArch64-MPAMHCR_EL2, AArch64-MPAMVPMV_EL2, and MPAMVPM<n>_EL2 with n in the range 0 to VPMR_MAX. Must be 0 if EL2 is not implemented in either Security state. **0b1** MPAM virtualization is supported.	0b1
[16]	**RES0**	Reserved	**RES0**
[15:0]	PARTID_MAX	The largest value of PARTID that the implementation can generate. The PARTID_I and PARTID_D fields of every MPAMn_ELx must implement at least enough bits to represent PARTID_MAX. **0b0000011111111111** Max PARTID field is 2047	0x07FF

Access

MRS <Xt>, MPAMIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0100	0b100

Accessibility

MRS <Xt>, MPAMIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAMHCR_EL2.TRAP_MPAMIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MPAM2_EL2.TIDR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        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.6.19 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-136: AArch64_imp_cpucfr_el1 bit assignments

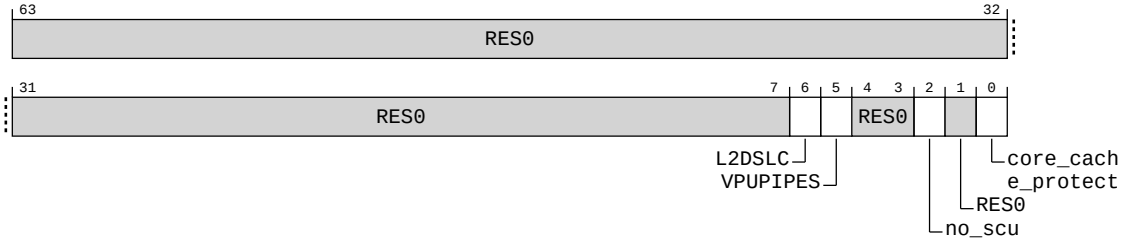


Table A-331: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:7]	**RES0**	Reserved	**RES0**
[6]	L2DSLC	L2 data RAM register slice presence. Possible values of this bit are: **0b0** No L2 data RAM register slice present **0b1** L2 data RAM register slice present	x
[5]	VPUPIPES	Indicates the number of Vector Processing Unit (VPU) pipes. Possible values of this bit are: **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.6.20 CLIDR_EL1, Cache Level ID Register

Identifies the type of cache, or caches, that are implemented at each level and can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx x000 0000 0001 0100 1100 0011 0000 0000 0000 0001 0010 0011
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-137: AArch64_clidr_el1 bit assignments

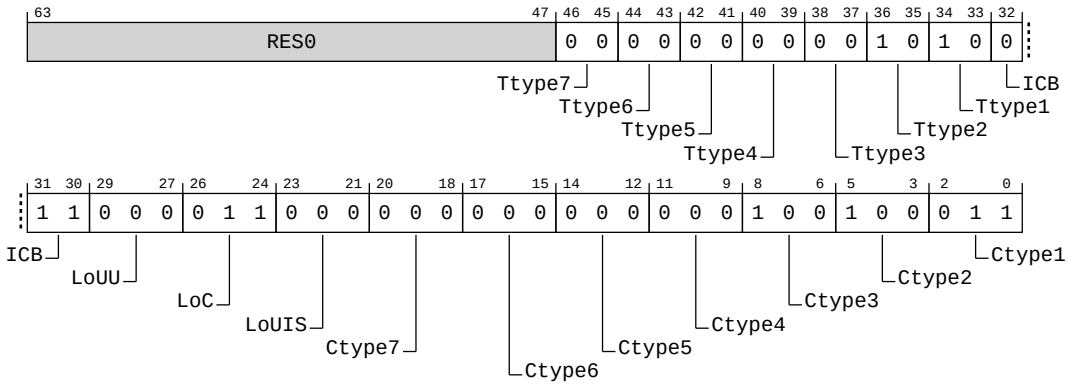


Table A-333: CLIDR_EL1 bit descriptions

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
[38:37]	Ttype3	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. **0b00** No Tag Cache.	0b00

Bits	Name	Description	Reset
[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. **0b011** L3 cache is the highest Inner Cacheable level.	0b011
[29:27]	LoUU	Level of Unification Uniprocessor for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **Note:** This field does not describe the requirements for instruction cache invalidation. See AArch64-CTR_EL0.DIC. **Note:** When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches. **0b000** Level of Unification Uniprocessor is before the L1 data cache.	0b000
[26:24]	LoC	Level of Coherence for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **0b011** Level 3	0b011
[23:21]	LoUIS	Level of Unification Inner Shareable for the cache hierarchy. For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions. **Note:** This field does not describe the requirements for instruction cache invalidation. See AArch64-CTR_EL0.DIC. **Note:** When FEAT_S2FWB is implemented, the architecture requires that this field is zero so that no levels of data cache need to be cleaned in order to manage coherency with instruction fetches. **0b000** No cache level needs cleaning to Point of Unification	0b000

Bits	Name	Description	Reset
[20:18]	Ctype7	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[17:15]	Ctype6	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[14:12]	Ctype5	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[11:9]	Ctype4	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b000** No cache.	0b000
[8:6]	Ctype3	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b100** Unified instruction and data caches at L3	0b100
[5:3]	Ctype2	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b100** Unified instruction and data caches at L2	0b100
[2:0]	Ctype1	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: **0b011** Separate instruction and data caches 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);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CLIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CLIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CLIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CLIDR_EL1;
```

A.6.21 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 0100
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-138: AArch64_gmid_el1 bit assignments

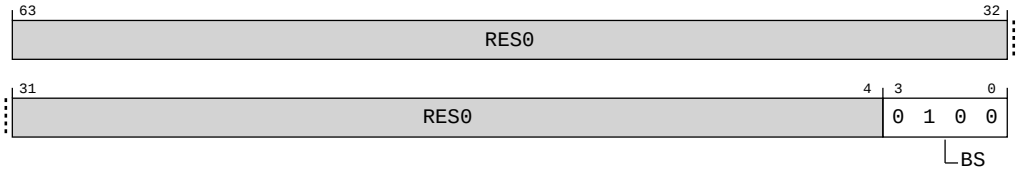


Table A-335: GMID_EL1 bit descriptions

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.6.22 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 xxx1 0100 0100 0100 11xx xxxx xxxx 0100
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-139: AArch64_ctr_el0 bit assignments

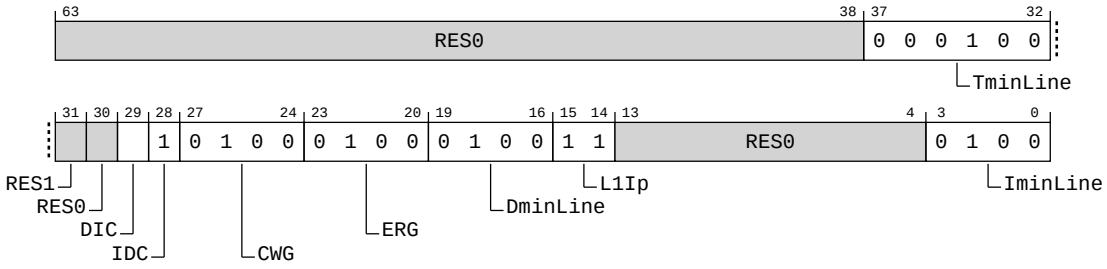


Table A-337: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:38]	**RES0**	Reserved	**RES0**
[37:32]	TminLine	Tag minimum Line. Log_2 of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE. **0b000100** Log2 of number of words (64/4=16) covered by Allocation Tags in the smallest cache line of all caches	0b000100
[31]	**RES1**	Reserved	**RES1**
[30]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[29]	DIC	Instruction cache invalidation requirements for data to instruction coherence. **0b0** When COHERENT_ICACHE not enabled, Instruction cache invalidation to the point of unification is required for instruction to data coherence. **0b1** When COHERENT_ICACHE enabled, Instruction cache cleaning to the point of unification is not required for instruction to data coherence.	The reset values can be the following: 0b0, 0b1, respective to the value.
[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' && SCR_EL3.FGTEn == '1' &&
 HFGRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.CTR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CTR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CTR_EL0;
```

A.6.23 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 xxx0 0100
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-140: AArch64_dczid_el0 bit assignments

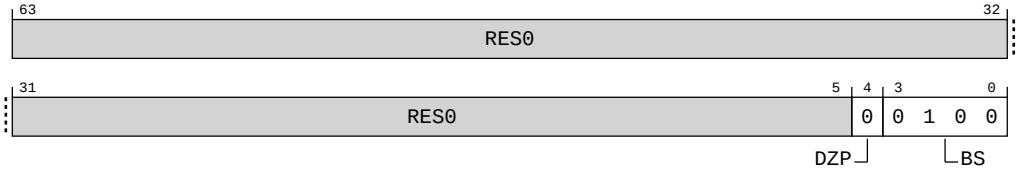


Table A-339: DCZID_EL0 bit descriptions

Bits	Name	Description	Reset
[63:5]	**RES0**	Reserved	**RES0**
[4]	DZP	Data Zero Prohibited. This field indicates whether use of DC ZVA instructions is permitted or prohibited. If FEAT_MTE is implemented, this field also indicates whether use of the DC GVA and DC GZVA instructions are permitted or prohibited. **0b0** Instructions are permitted.	0b0
[3:0]	BS	Log_2 of the block size in words. The maximum size supported is 2KB, indicated by value 0b1001. If FEAT_MTE2 is implemented, the minimum size supported is 16 bytes, indicated by value 0b0010. **0b0100** 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
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    else
        X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = DCZID_EL0;
```

A.7 AArch64 Special-purpose registers summary

The following summary table provides an overview of all Special-purpose registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-341: Special-purpose registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SPSR_EL1	3	0	C4	C0	0	See individual bit resets.	64-bit	Saved Program Status Register (EL1)
ELR_EL1	3	0	C4	C0	1	See individual bit resets.	64-bit	Exception Link Register (EL1)
SP_EL0	3	0	C4	C1	0	See individual bit resets.	64-bit	Stack Pointer (EL0)
DSPSR_EL0	3	3	C4	C5	0	See individual bit resets.	64-bit	Debug Saved Program Status Register
DLR_EL0	3	3	C4	C5	1	See individual bit resets.	64-bit	Debug Link Register
SPSR_EL2	3	4	C4	C0	0	See individual bit resets.	64-bit	Saved Program Status Register (EL2)
ELR_EL2	3	4	C4	C0	1	See individual bit resets.	64-bit	Exception Link Register (EL2)
SP_EL1	3	4	C4	C1	0	See individual bit resets.	64-bit	Stack Pointer (EL1)
SPSR_irq	3	4	C4	C3	0	See individual bit resets.	64-bit	Saved Program Status Register (IRQ mode)
SPSR_abt	3	4	C4	C3	1	See individual bit resets.	64-bit	Saved Program Status Register (Abort mode)
SPSR_und	3	4	C4	C3	2	See individual bit resets.	64-bit	Saved Program Status Register (Undefined mode)
SPSR_fiq	3	4	C4	C3	3	See individual bit resets.	64-bit	Saved Program Status Register (FIQ mode)
SPSR_EL3	3	6	C4	C0	0	See individual bit resets.	64-bit	Saved Program Status Register (EL3)
ELR_EL3	3	6	C4	C0	1	See individual bit resets.	64-bit	Exception Link Register (EL3)
SP_EL2	3	6	C4	C1	0	See individual bit resets.	64-bit	Stack Pointer (EL2)

A.8 AArch64 Performance Monitors registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-342: Performance Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMINTENSET_EL1	3	0	C9	C14	1	See individual bit resets.	64-bit	Performance Monitors Interrupt Enable Set register
PMINTENCLR_EL1	3	0	C9	C14	2	See individual bit resets.	64-bit	Performance Monitors Interrupt Enable Clear register
PMMIR_EL1	3	0	C9	C14	6	See individual bit resets.	64-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	3	C9	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register
PMCNTENSET_EL0	3	3	C9	C12	1	See individual bit resets.	64-bit	Performance Monitors Count Enable Set register
PMCNTENCLR_EL0	3	3	C9	C12	2	See individual bit resets.	64-bit	Performance Monitors Count Enable Clear register
PMOVSCLR_EL0	3	3	C9	C12	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_EL0	3	3	C9	C12	4	See individual bit resets.	64-bit	Performance Monitors Software Increment register
PMSELR_EL0	3	3	C9	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	3	3	C9	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	3	C9	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 1
PMCCNTR_EL0	3	3	C9	C13	0	See individual bit resets.	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	3	3	C9	C13	1	See individual bit resets.	64-bit	Performance Monitors Selected Event Type Register
PMXEVCNTR_EL0	3	3	C9	C13	2	See individual bit resets.	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	3	3	C9	C14	0	See individual bit resets.	64-bit	Performance Monitors User Enable Register
PMOVSSET_EL0	3	3	C9	C14	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Set register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMEVCNTR0_EL0	3	3	C14	C8	0	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	3	3	C14	C8	1	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR2_EL0	3	3	C14	C8	2	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR3_EL0	3	3	C14	C8	3	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR4_EL0	3	3	C14	C8	4	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVCNTR5_EL0	3	3	C14	C8	5	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVTYPER0_EL0	3	3	C14	C12	0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	3	3	C14	C12	1	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER2_EL0	3	3	C14	C12	2	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER3_EL0	3	3	C14	C12	3	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER4_EL0	3	3	C14	C12	4	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMEVTYPER5_EL0	3	3	C14	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_EL0	3	3	C14	C15	7	See individual bit resets.	64-bit	Performance Monitors Cycle Count Filter Register

A.8.1 PMMIR_EL1, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation to software.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 0000 0000 1010
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
```

```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-141: AArch64_pmmir_el1 bit assignments

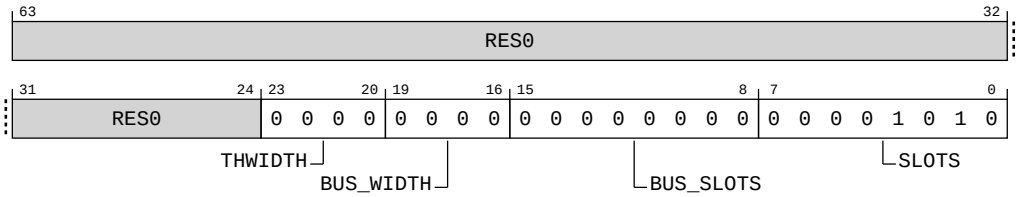


Table A-343: PMMIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**
[23:20]	THWIDTH	AArch64-PMEVTYPER<n>_EL0.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the AArch64-PMEVTYPER<n>_EL0.TH field. **0b0000** FEAT_PMUv3_TH is not implemented.	0b0000
[19:16]	BUS_WIDTH	Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as Log_2(number of bytes), plus one. **0b0000** The information is not available.	0b0000
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. **0b00000000** The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is 0	0x00
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. **0b00001010** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 10.	0x0A

Access

MRS <Xt>, PMMIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b110

Accessibility

MRS <Xt>, PMMIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMMIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.8.2 PMCR_EL0, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

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 0000 0000 xxxx xxxx 0011 0xxx xxx0 x000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-142: AArch64_pmcr_el0 bit assignments

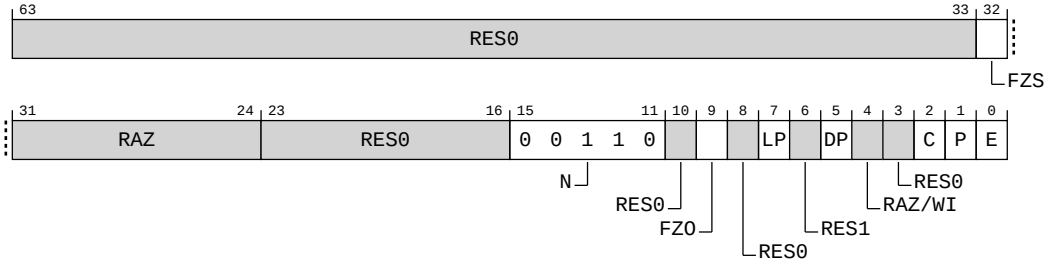


Table A-345: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:33]	**RES0**	Reserved	**RES0**
[32]	FZS	Freeze-on-SPE event. Stop counters when AArch64-PMBLIMITR_EL1.{PMFZ,E} == {1,1} and AArch64-PMBSR_EL1.S == 1. In the description of this field: • If EL2 is implemented, then PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, then PMN is PMCR_EL0.N. **0b0** Do not freeze on Statistical Profiling Buffer Management event.	x
[31:24]	**RAZ**	Reserved	**RAZ**
[23:16]	**RES0**	Reserved	**RES0**
[15:11]	N	Number of event counters: **0b00110** 6 PMU counters implemented	0b00110
[10]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[9]	FZO	Freeze-on-overflow. Stop event counters on overflow. In the description of this field: • If EL2 is implemented, then PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, then PMN is PMCR_EL0.N. **0b0** Do not freeze on overflow.	x
[8]	**RES0**	Reserved	**RES0**
[7]	LP	Long event counter enable. Determines which event counter bit generates an overflow recorded by AArch32-PMOVSR[n]. In the description of this field: • If EL2 is implemented, then PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, then PMN is PMCR_EL0.N. **0b0** Affected counters overflow on unsigned overflow of AArch64-PMEVCNTR<n>_EL0[31:0].	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 FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and AArch64-MDCR_EL3.MPMX is 1, then cycle counting by AArch64-PMCCNTR_EL0 is disabled at EL3. • If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZO, then cycle counting by AArch64-PMCCNTR_EL0 is disabled. • If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZS, then it is IMPLEMENTATION DEFINED whether cycle counting by AArch64-PMCCNTR_EL0 is disabled. • 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.	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. 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. Access to this field is: WO/**RAZ**	0b0
[0]	E	Enable. In the description of this field: • If EL2 is implemented, then PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, then PMN is PMCR_EL0.N. **0b0** Affected counters are disabled and do not count. **0b1** Affected counters are enabled by AArch64-PMCNTENSET_EL0.	0b0

Access

MRS <Xt>, PMCR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

MSR PMCR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

Accessibility

MRS <Xt>, PMCR_EL0

```
if PSTATE.EL == EL0 then
    if 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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMCR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMCR_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        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.8.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

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 0110 1111 0011 1111

Bit descriptions

Figure A-143: AArch64_pmceid0_el0 bit assignments

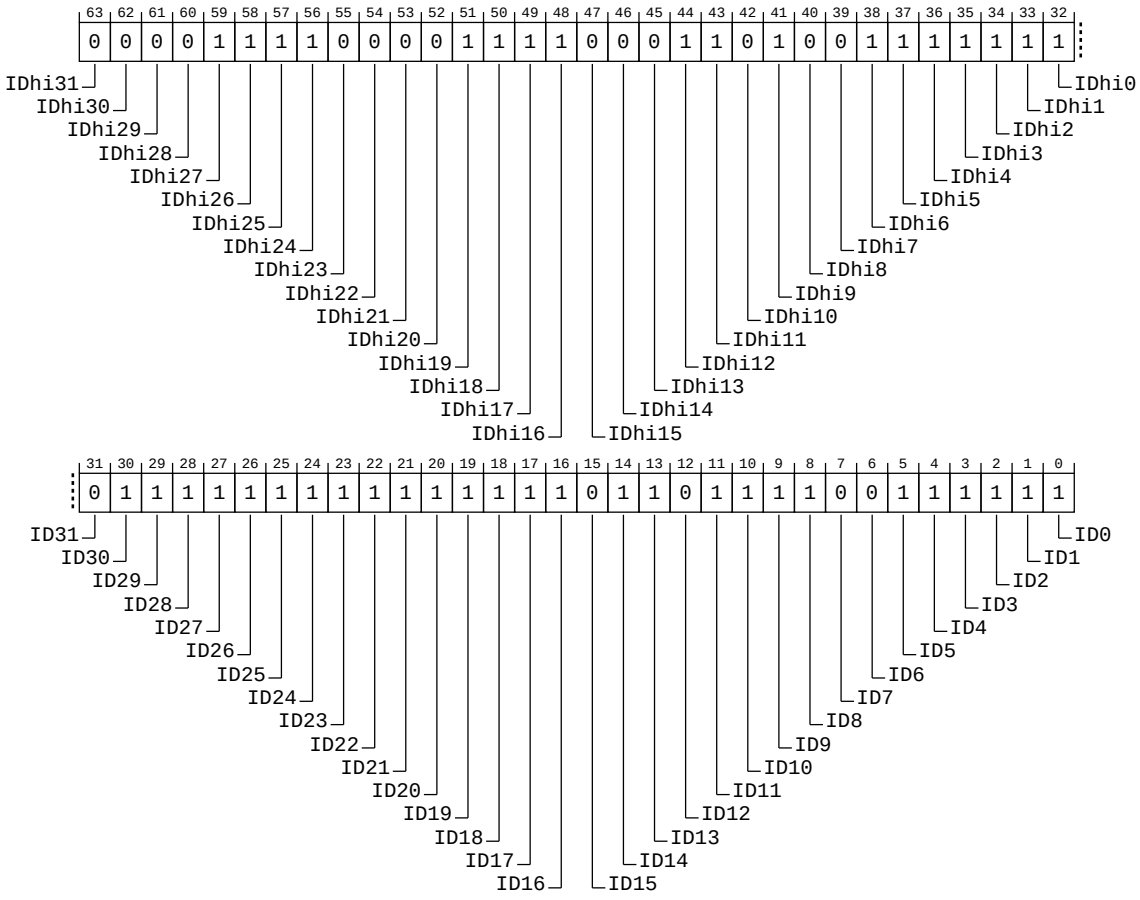


Table A-348: PMCEID0_EL0 bit descriptions

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 **0b1** The Common event is implemented.	0b1
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **0b1** The Common event is implemented.	0b1
[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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID0_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.8.4 PMCEID1_EL0, Performance Monitors Common Event Identification register 1

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0020 to 0x003F and 0x4020 to 0x403F.

For more information about the Common events and the use of the PMCEID<n>_EL0 registers see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

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-144: AArch64_pmceid1_el0 bit assignments

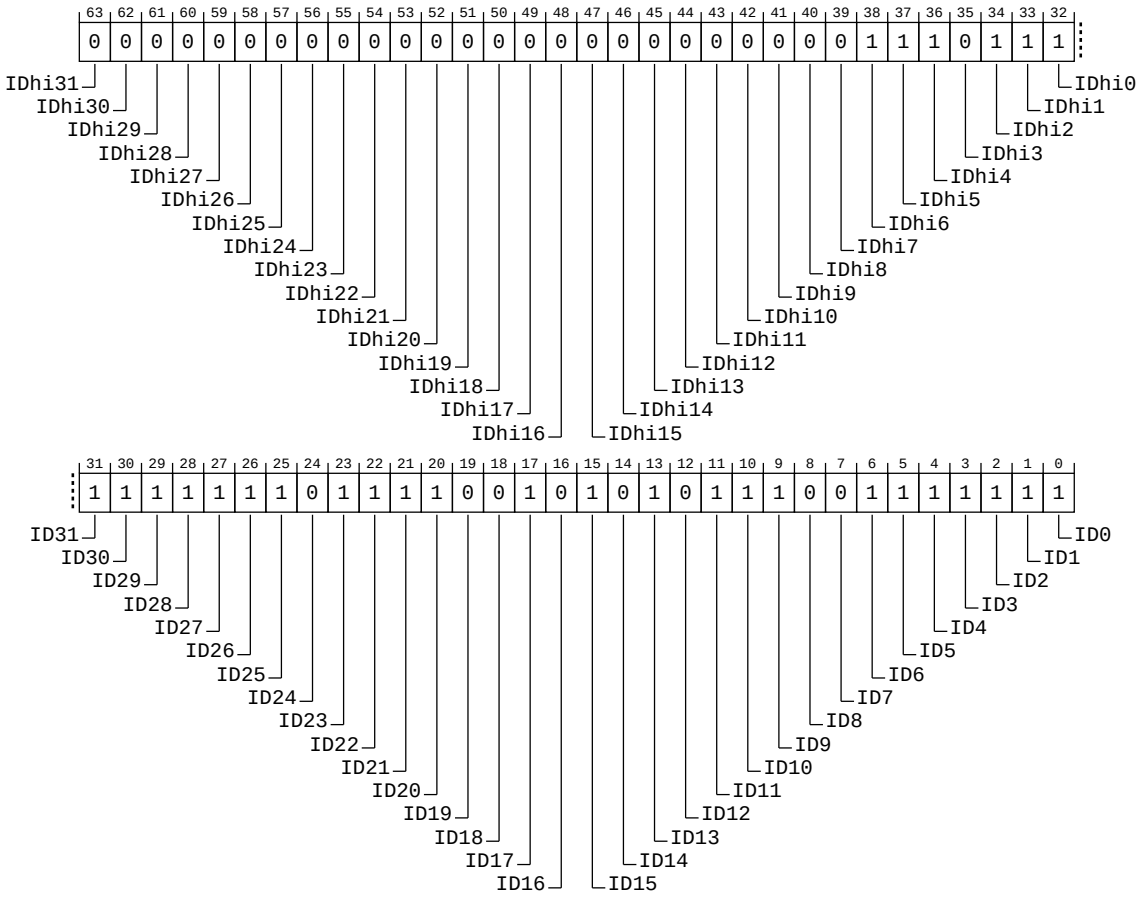


Table A-350: PMCEID1_EL0 bit descriptions

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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCEID1_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMCEIDn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.8.5 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-145: AArch64_pmevcntr0_el0 bit assignments

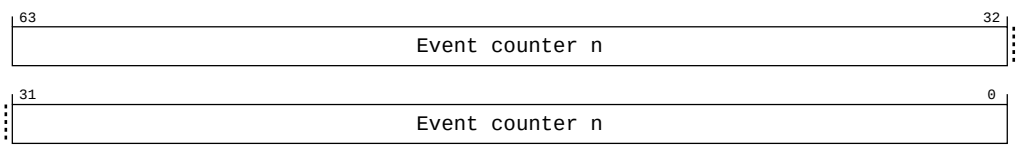


Table A-352: PMEVCNTR0_EL0 bit descriptions

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVCNTR_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.6 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-146: AArch64_pmevcntr1_el0 bit assignments

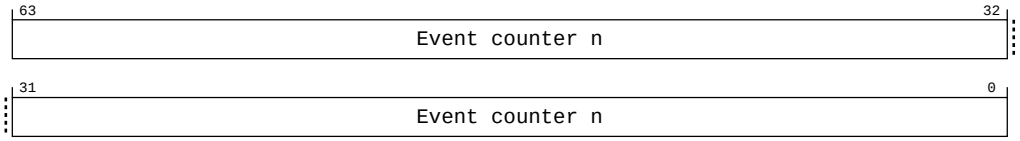


Table A-355: PMEVCNTR1_EL0 bit descriptions

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVCNTR_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.7 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-147: AArch64_pmevcntr2_el0 bit assignments

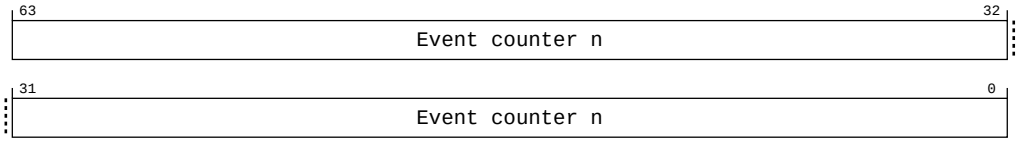


Table A-358: PMEVCNTR2_EL0 bit descriptions

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVCNTR_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.8 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-148: AArch64_pmevcntr3_el0 bit assignments

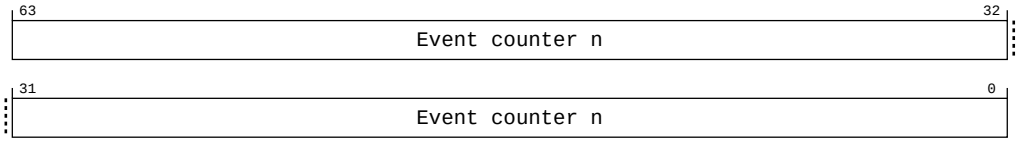


Table A-361: PMEVCNTR3_EL0 bit descriptions

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVCNTR_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.9 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-149: AArch64_pmevcntr4_el0 bit assignments

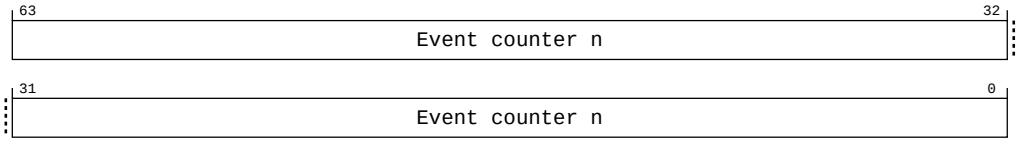


Table A-364: PMEVCNTR4_EL0 bit descriptions

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVCNTR_EL0[4];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.10 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-150: AArch64_pmevcntr5_el0 bit assignments

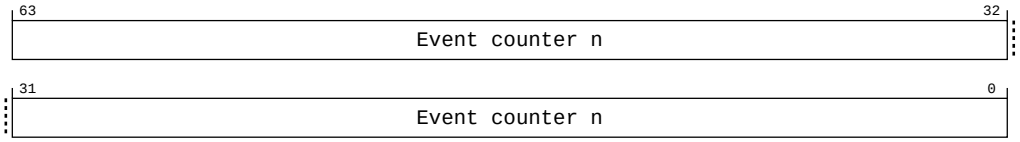


Table A-367: PMEVCNTR5_EL0 bit descriptions

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVCNTR<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVCNTR_EL0[5];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVCNTRn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.11 PMEVTYPER0_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-151: AArch64_pmevtyper0_el0 bit assignments

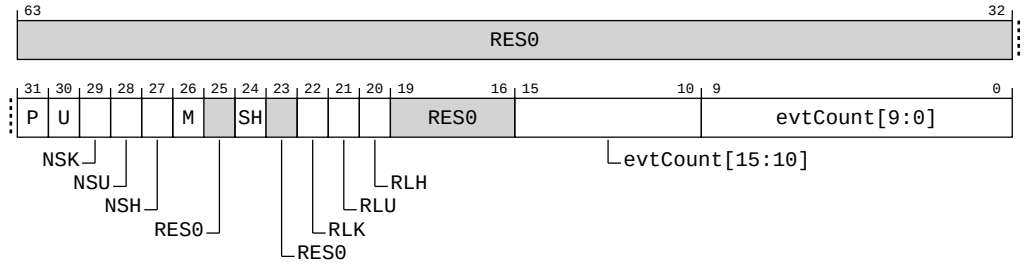


Table A-370: PMEVTYPER0_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P	EL1 filtering. Controls counting events in EL1. **0b0** This field has no effect on filtering of events. **0b1** Events in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting events in EL0. **0b0** This field has no effect on filtering of events. **0b1** Events in EL0 are not counted.	x
[29]	NSK	Non-secure EL1 filtering. Controls counting events in Non-secure EL1. If PMEVTYPER<n>_EL0.NSK is not equal to PMEVTYPER<n>_EL0.P, then events in Non-secure EL1 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSK has no effect on filtering of events in Non-secure EL1. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in Non-secure EL1 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in Non-secure EL1 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[28]	NSU	Non-secure EL0 filtering. Controls counting events in Non-secure EL0. If PMEVTYPER<n>_EL0.NSU is not equal to PMEVTYPER<n>_EL0.U, then events in Non-secure EL0 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSU has no effect on filtering of events in Non-secure EL0. **0b0** When PMEVTYPER<n>_EL0.U == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.U == 1, events in Non-secure EL0 are not counted. **0b1** When PMEVTYPER<n>_EL0.U == 0, events in Non-secure EL0 are not counted. When PMEVTYPER<n>_EL0.U == 1, this field has no effect on filtering of events.	x
[27]	NSH	EL2 filtering. Controls counting events in EL2. **0b0** Events in EL2 are not counted. **0b1** This field has no effect on filtering of events.	x

Bits	Name	Description	Reset
[26]	M	EL3 filtering. Controls counting events in EL3. If PMEVTYPER<n>_EL0.M is not equal to PMEVTYPER<n>_EL0.P, then events in EL3 are not counted. Otherwise, PMEVTYPER<n>_EL0.M has no effect on filtering of events in EL3. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in EL3 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in EL3 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. Controls counting events in Secure EL2. If PMEVTYPER<n>_EL0.SH is equal to PMEVTYPER<n>_EL0.NSH, then events in Secure EL2 are not counted. Otherwise, PMEVTYPER<n>_EL0.SH has no effect on filtering of events in Secure EL2. **0b0** When PMEVTYPER<n>_EL0.NSH == 0, events in Secure EL2 are not counted. When PMEVTYPER<n>_EL0.NSH == 1, this field has no effect on filtering of events. **0b1** When PMEVTYPER<n>_EL0.NSH == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.NSH == 1, events in Secure EL2 are not counted.	x
[23]	**RES0**	Reserved	**RES0**
[22]	RLK	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.	xxxx
[21]	RLU	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.	xxxx
[20]	RLH	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL2 filtering bit. Controls counting in Realm EL2. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.	xxxx
[19:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVTYPER_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 0 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.12 PMEVTYPER1_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-152: AArch64_pmevtyper1_el0 bit assignments

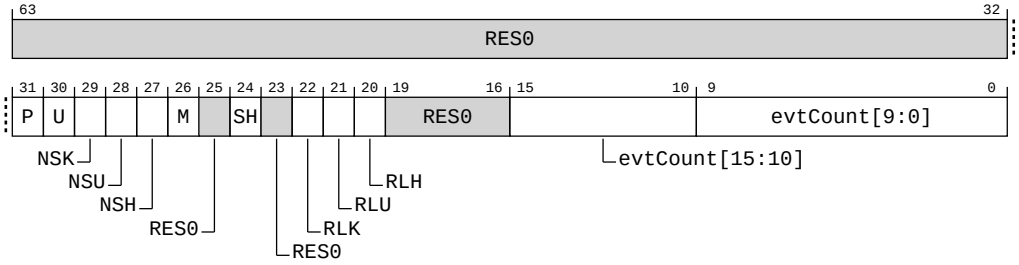


Table A-373: PMEVTYPER1_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	P	EL1 filtering. Controls counting events in EL1. **0b0** This field has no effect on filtering of events. **0b1** Events in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting events in EL0. **0b0** This field has no effect on filtering of events. **0b1** Events in EL0 are not counted.	x
[29]	NSK	Non-secure EL1 filtering. Controls counting events in Non-secure EL1. If PMEVTYPER<n>_EL0.NSK is not equal to PMEVTYPER<n>_EL0.P, then events in Non-secure EL1 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSK has no effect on filtering of events in Non-secure EL1. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in Non-secure EL1 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in Non-secure EL1 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 filtering. Controls counting events in Non-secure EL0. If PMEVTYPER<n>_EL0.NSU is not equal to PMEVTYPER<n>_EL0.U, then events in Non-secure EL0 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSU has no effect on filtering of events in Non-secure EL0. **0b0** When PMEVTYPER<n>_EL0.U == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.U == 1, events in Non-secure EL0 are not counted. **0b1** When PMEVTYPER<n>_EL0.U == 0, events in Non-secure EL0 are not counted. When PMEVTYPER<n>_EL0.U == 1, this field has no effect on filtering of events.	x
[27]	NSH	EL2 filtering. Controls counting events in EL2. **0b0** Events in EL2 are not counted. **0b1** This field has no effect on filtering of events.	x
[26]	M	EL3 filtering. Controls counting events in EL3. If PMEVTYPER<n>_EL0.M is not equal to PMEVTYPER<n>_EL0.P, then events in EL3 are not counted. Otherwise, PMEVTYPER<n>_EL0.M has no effect on filtering of events in EL3. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in EL3 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in EL3 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. Controls counting events in Secure EL2. If PMEVTYPER<n>_EL0.SH is equal to PMEVTYPER<n>_EL0.NSH, then events in Secure EL2 are not counted. Otherwise, PMEVTYPER<n>_EL0.SH has no effect on filtering of events in Secure EL2. **0b0** When PMEVTYPER<n>_EL0.NSH == 0, events in Secure EL2 are not counted. When PMEVTYPER<n>_EL0.NSH == 1, this field has no effect on filtering of events. **0b1** When PMEVTYPER<n>_EL0.NSH == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.NSH == 1, events in Secure EL2 are not counted.	x
[23]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[22]	RLK	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.	xxxx
[21]	RLU	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.	xxxx
[20]	RLH	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL2 filtering bit. Controls counting in Realm EL2. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.	xxxx
[19:16]	**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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVTYPER_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 1 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.13 PMEVTYPER2_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-153: AArch64_pmevtyper2_el0 bit assignments

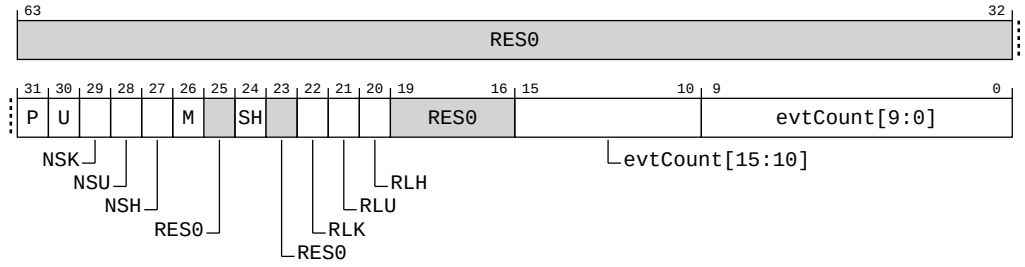


Table A-376: PMEVTYPER2_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P	EL1 filtering. Controls counting events in EL1. **0b0** This field has no effect on filtering of events. **0b1** Events in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting events in EL0. **0b0** This field has no effect on filtering of events. **0b1** Events in EL0 are not counted.	x
[29]	NSK	Non-secure EL1 filtering. Controls counting events in Non-secure EL1. If PMEVTYPER<n>_EL0.NSK is not equal to PMEVTYPER<n>_EL0.P, then events in Non-secure EL1 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSK has no effect on filtering of events in Non-secure EL1. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in Non-secure EL1 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in Non-secure EL1 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[28]	NSU	Non-secure EL0 filtering. Controls counting events in Non-secure EL0. If PMEVTYPER<n>_EL0.NSU is not equal to PMEVTYPER<n>_EL0.U, then events in Non-secure EL0 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSU has no effect on filtering of events in Non-secure EL0. **0b0** When PMEVTYPER<n>_EL0.U == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.U == 1, events in Non-secure EL0 are not counted. **0b1** When PMEVTYPER<n>_EL0.U == 0, events in Non-secure EL0 are not counted. When PMEVTYPER<n>_EL0.U == 1, this field has no effect on filtering of events.	x
[27]	NSH	EL2 filtering. Controls counting events in EL2. **0b0** Events in EL2 are not counted. **0b1** This field has no effect on filtering of events.	x

Bits	Name	Description	Reset
[26]	M	EL3 filtering. Controls counting events in EL3. If PMEVTYPER<n>_EL0.M is not equal to PMEVTYPER<n>_EL0.P, then events in EL3 are not counted. Otherwise, PMEVTYPER<n>_EL0.M has no effect on filtering of events in EL3. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in EL3 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in EL3 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. Controls counting events in Secure EL2. If PMEVTYPER<n>_EL0.SH is equal to PMEVTYPER<n>_EL0.NSH, then events in Secure EL2 are not counted. Otherwise, PMEVTYPER<n>_EL0.SH has no effect on filtering of events in Secure EL2. **0b0** When PMEVTYPER<n>_EL0.NSH == 0, events in Secure EL2 are not counted. When PMEVTYPER<n>_EL0.NSH == 1, this field has no effect on filtering of events. **0b1** When PMEVTYPER<n>_EL0.NSH == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.NSH == 1, events in Secure EL2 are not counted.	x
[23]	**RES0**	Reserved	**RES0**
[22]	RLK	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.	xxxx
[21]	RLU	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.	xxxx
[20]	RLH	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL2 filtering bit. Controls counting in Realm EL2. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.	xxxx
[19:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVTYPER_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 2 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.14 PMEVTYPER3_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-154: AArch64_pmevtyper3_el0 bit assignments

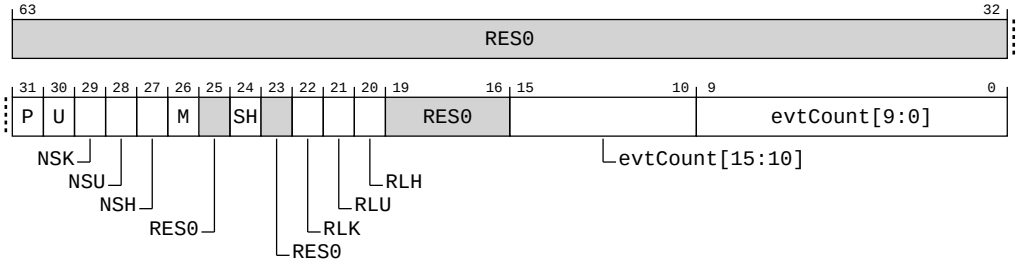


Table A-379: PMEVTYPER3_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	P	EL1 filtering. Controls counting events in EL1. **0b0** This field has no effect on filtering of events. **0b1** Events in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting events in EL0. **0b0** This field has no effect on filtering of events. **0b1** Events in EL0 are not counted.	x
[29]	NSK	Non-secure EL1 filtering. Controls counting events in Non-secure EL1. If PMEVTYPER<n>_EL0.NSK is not equal to PMEVTYPER<n>_EL0.P, then events in Non-secure EL1 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSK has no effect on filtering of events in Non-secure EL1. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in Non-secure EL1 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in Non-secure EL1 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 filtering. Controls counting events in Non-secure EL0. If PMEVTYPER<n>_EL0.NSU is not equal to PMEVTYPER<n>_EL0.U, then events in Non-secure EL0 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSU has no effect on filtering of events in Non-secure EL0. **0b0** When PMEVTYPER<n>_EL0.U == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.U == 1, events in Non-secure EL0 are not counted. **0b1** When PMEVTYPER<n>_EL0.U == 0, events in Non-secure EL0 are not counted. When PMEVTYPER<n>_EL0.U == 1, this field has no effect on filtering of events.	x
[27]	NSH	EL2 filtering. Controls counting events in EL2. **0b0** Events in EL2 are not counted. **0b1** This field has no effect on filtering of events.	x
[26]	M	EL3 filtering. Controls counting events in EL3. If PMEVTYPER<n>_EL0.M is not equal to PMEVTYPER<n>_EL0.P, then events in EL3 are not counted. Otherwise, PMEVTYPER<n>_EL0.M has no effect on filtering of events in EL3. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in EL3 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in EL3 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. Controls counting events in Secure EL2. If PMEVTYPER<n>_EL0.SH is equal to PMEVTYPER<n>_EL0.NSH, then events in Secure EL2 are not counted. Otherwise, PMEVTYPER<n>_EL0.SH has no effect on filtering of events in Secure EL2. **0b0** When PMEVTYPER<n>_EL0.NSH == 0, events in Secure EL2 are not counted. When PMEVTYPER<n>_EL0.NSH == 1, this field has no effect on filtering of events. **0b1** When PMEVTYPER<n>_EL0.NSH == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.NSH == 1, events in Secure EL2 are not counted.	x
[23]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[22]	RLK	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.	xxxx
[21]	RLU	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.	xxxx
[20]	RLH	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL2 filtering bit. Controls counting in Realm EL2. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.	xxxx
[19:16]	**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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVTYPER_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 3 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.15 PMEVTYPER4_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-155: AArch64_pmevtyper4_el0 bit assignments

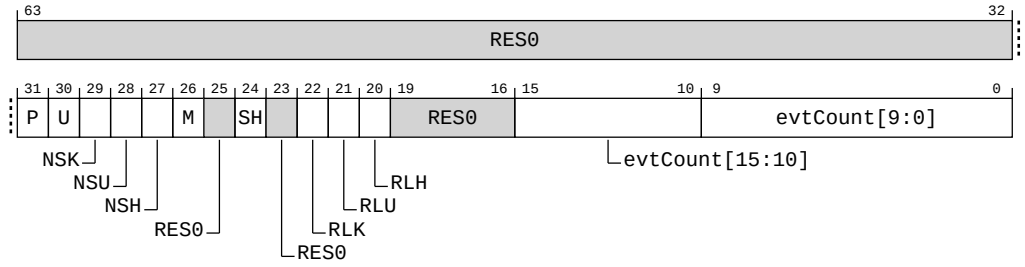


Table A-382: PMEVTYPER4_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	P	EL1 filtering. Controls counting events in EL1. **0b0** This field has no effect on filtering of events. **0b1** Events in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting events in EL0. **0b0** This field has no effect on filtering of events. **0b1** Events in EL0 are not counted.	x
[29]	NSK	Non-secure EL1 filtering. Controls counting events in Non-secure EL1. If PMEVTYPER<n>_EL0.NSK is not equal to PMEVTYPER<n>_EL0.P, then events in Non-secure EL1 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSK has no effect on filtering of events in Non-secure EL1. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in Non-secure EL1 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in Non-secure EL1 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[28]	NSU	Non-secure EL0 filtering. Controls counting events in Non-secure EL0. If PMEVTYPER<n>_EL0.NSU is not equal to PMEVTYPER<n>_EL0.U, then events in Non-secure EL0 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSU has no effect on filtering of events in Non-secure EL0. **0b0** When PMEVTYPER<n>_EL0.U == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.U == 1, events in Non-secure EL0 are not counted. **0b1** When PMEVTYPER<n>_EL0.U == 0, events in Non-secure EL0 are not counted. When PMEVTYPER<n>_EL0.U == 1, this field has no effect on filtering of events.	x
[27]	NSH	EL2 filtering. Controls counting events in EL2. **0b0** Events in EL2 are not counted. **0b1** This field has no effect on filtering of events.	x

Bits	Name	Description	Reset
[26]	M	EL3 filtering. Controls counting events in EL3. If PMEVTYPER<n>_EL0.M is not equal to PMEVTYPER<n>_EL0.P, then events in EL3 are not counted. Otherwise, PMEVTYPER<n>_EL0.M has no effect on filtering of events in EL3. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in EL3 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in EL3 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. Controls counting events in Secure EL2. If PMEVTYPER<n>_EL0.SH is equal to PMEVTYPER<n>_EL0.NSH, then events in Secure EL2 are not counted. Otherwise, PMEVTYPER<n>_EL0.SH has no effect on filtering of events in Secure EL2. **0b0** When PMEVTYPER<n>_EL0.NSH == 0, events in Secure EL2 are not counted. When PMEVTYPER<n>_EL0.NSH == 1, this field has no effect on filtering of events. **0b1** When PMEVTYPER<n>_EL0.NSH == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.NSH == 1, events in Secure EL2 are not counted.	x
[23]	**RES0**	Reserved	**RES0**
[22]	RLK	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.	xxxx
[21]	RLU	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.	xxxx
[20]	RLH	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL2 filtering bit. Controls counting in Realm EL2. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.	xxxx
[19:16]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVTYPER_EL0[4];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 4 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.8.16 PMEVTYPER5_EL0, Performance Monitors Event Type Registers

Configures event counter n, where n is 0 to 30.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-156: AArch64_pmevtyper5_el0 bit assignments

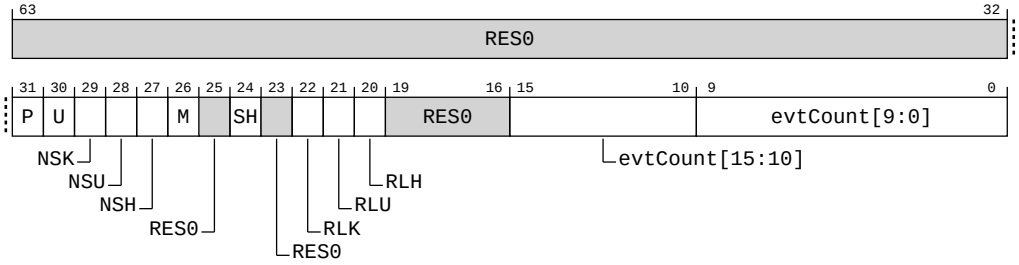


Table A-385: PMEVTYPER5_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	P	EL1 filtering. Controls counting events in EL1. **0b0** This field has no effect on filtering of events. **0b1** Events in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting events in EL0. **0b0** This field has no effect on filtering of events. **0b1** Events in EL0 are not counted.	x
[29]	NSK	Non-secure EL1 filtering. Controls counting events in Non-secure EL1. If PMEVTYPER<n>_EL0.NSK is not equal to PMEVTYPER<n>_EL0.P, then events in Non-secure EL1 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSK has no effect on filtering of events in Non-secure EL1. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in Non-secure EL1 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in Non-secure EL1 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x

Bits	Name	Description	Reset
[28]	NSU	Non-secure EL0 filtering. Controls counting events in Non-secure EL0. If PMEVTYPER<n>_EL0.NSU is not equal to PMEVTYPER<n>_EL0.U, then events in Non-secure EL0 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSU has no effect on filtering of events in Non-secure EL0. **0b0** When PMEVTYPER<n>_EL0.U == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.U == 1, events in Non-secure EL0 are not counted. **0b1** When PMEVTYPER<n>_EL0.U == 0, events in Non-secure EL0 are not counted. When PMEVTYPER<n>_EL0.U == 1, this field has no effect on filtering of events.	x
[27]	NSH	EL2 filtering. Controls counting events in EL2. **0b0** Events in EL2 are not counted. **0b1** This field has no effect on filtering of events.	x
[26]	M	EL3 filtering. Controls counting events in EL3. If PMEVTYPER<n>_EL0.M is not equal to PMEVTYPER<n>_EL0.P, then events in EL3 are not counted. Otherwise, PMEVTYPER<n>_EL0.M has no effect on filtering of events in EL3. **0b0** When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in EL3 are not counted. **0b1** When PMEVTYPER<n>_EL0.P == 0, events in EL3 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.	x
[25]	**RES0**	Reserved	**RES0**
[24]	SH	Secure EL2 filtering. Controls counting events in Secure EL2. If PMEVTYPER<n>_EL0.SH is equal to PMEVTYPER<n>_EL0.NSH, then events in Secure EL2 are not counted. Otherwise, PMEVTYPER<n>_EL0.SH has no effect on filtering of events in Secure EL2. **0b0** When PMEVTYPER<n>_EL0.NSH == 0, events in Secure EL2 are not counted. When PMEVTYPER<n>_EL0.NSH == 1, this field has no effect on filtering of events. **0b1** When PMEVTYPER<n>_EL0.NSH == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.NSH == 1, events in Secure EL2 are not counted.	x
[23]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[22]	RLK	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted. Otherwise, events in Realm EL1 are not counted.	xxxx
[21]	RLU	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0. If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted. Otherwise, events in Realm EL0 are not counted.	xxxx
[20]	RLH	**Otherwise** RES0 **When IsFeatureImplemented(FEAT_RME)** Realm EL2 filtering bit. Controls counting in Realm EL2. If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted. Otherwise, events in Realm EL2 are not counted.	xxxx
[19:16]	**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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is **UNDEFINED**.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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 FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of AArch64-PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is UNDEFINED.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <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.

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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() 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] = PMEVTYPER_EL0[5];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() 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] = 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
    UNDEFINED;
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() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HDFGWTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMEVTYPERn_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && 5 >= AArch64.GetNumEventCountersAccessible() then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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.9 AArch64 GIC system registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-388: GIC system registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ICV_RPR_EL1	3	0	C12	C11	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Running Priority Register
ICC_SGI1R_EL1	3	0	C12	C11	5	See individual bit resets.	64-bit	Interrupt Controller Software Generated Interrupt Group 1 Register
ICC_ASGI1R_EL1	3	0	C12	C11	6	See individual bit resets.	64-bit	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
ICC_SGI0R_EL1	3	0	C12	C11	7	See individual bit resets.	64-bit	Interrupt Controller Software Generated Interrupt Group 0 Register
ICC_IAR1_EL1	3	0	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller Interrupt Acknowledge Register 1
ICV_IAR1_EL1	3	0	C12	C12	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Acknowledge Register 1
ICC_EOIR1_EL1	3	0	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller End Of Interrupt Register 1
ICV_EOIR1_EL1	3	0	C12	C12	1	See individual bit resets.	64-bit	Interrupt Controller Virtual End Of Interrupt Register 1
ICC_HPPIR1_EL1	3	0	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 1
ICV_HPPIR1_EL1	3	0	C12	C12	2	See individual bit resets.	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
ICC_BPR1_EL1	3	0	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller Binary Point Register 1
ICV_BPR1_EL1	3	0	C12	C12	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Binary Point Register 1
ICC_CTLR_EL1	3	0	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	0	C12	C12	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Control Register
ICC_SRE_EL1	3	0	C12	C12	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable register (EL1)
ICC_IGRPEN0_EL1	3	0	C12	C12	6	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 0 Enable register
ICV_IGRPEN0_EL1	3	0	C12	C12	6	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 0 Enable register
ICC_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Interrupt Group 1 Enable register
ICV_IGRPEN1_EL1	3	0	C12	C12	7	See individual bit resets.	64-bit	Interrupt Controller Virtual Interrupt Group 1 Enable register
ICH_AP0R0_EL2	3	4	C12	C8	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R0_EL2	3	4	C12	C9	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 1 Registers
ICC_SRE_EL2	3	4	C12	C9	5	See individual bit resets.	64-bit	Interrupt Controller System Register Enable register (EL2)
ICH_HCR_EL2	3	4	C12	C11	0	See individual bit resets.	64-bit	Interrupt Controller Hyp Control Register

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

A.9.1 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

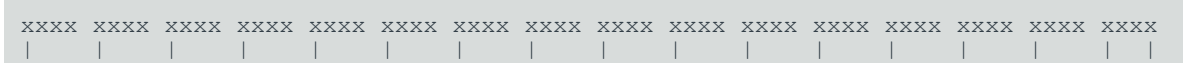
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-157: AArch64_icc_ap0r0_el1 bit assignments

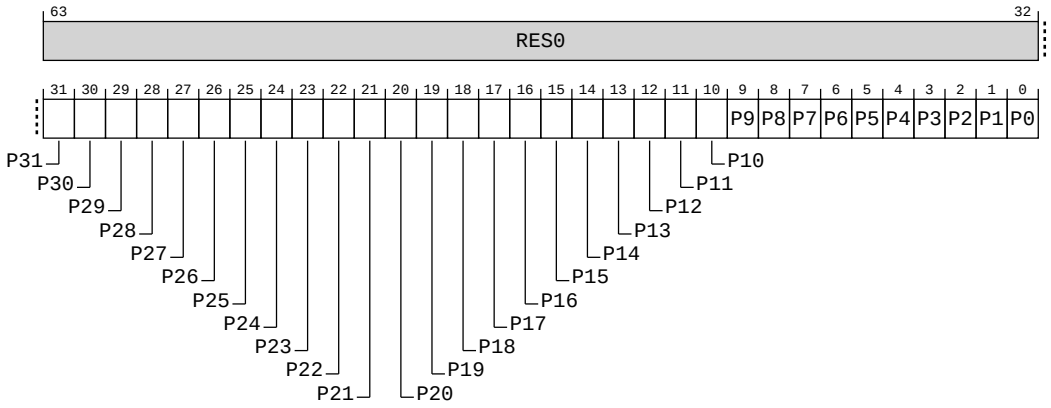


Table A-389: ICC_AP0R0_EL1 bit descriptions

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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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 implemented only in implementations that support 6 or more bits of priority. ICC_AP0R2_EL1 and ICC_AP0R3_EL1 are implemented only 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 implemented only in implementations that support 6 or more bits of priority. ICC_AP0R2_EL1 and ICC_AP0R3_EL1 are implemented only 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.9.2 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-158: AArch64_icv_ap0r0_el1 bit assignments

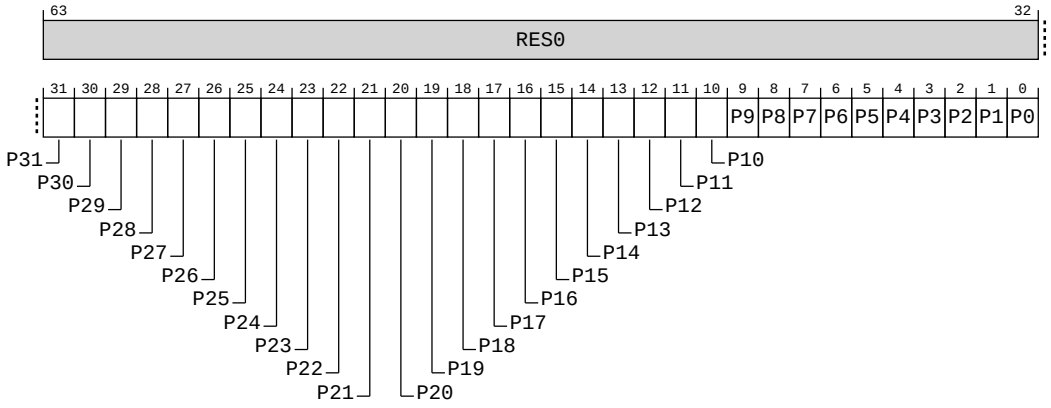


Table A-392: ICV_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[30]	P30	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[29]	P29	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[28]	P28	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[27]	P27	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[26]	P26	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[25]	P25	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[24]	P24	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[23]	P23	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[22]	P22	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[21]	P21	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[20]	P20	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[19]	P19	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[18]	P18	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[17]	P17	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[16]	P16	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[15]	P15	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[14]	P14	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[13]	P13	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[12]	P12	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[11]	P11	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[10]	P10	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[9]	P9	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[8]	P8	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[7]	P7	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[6]	P6	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[5]	P5	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[4]	P4	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

Bits	Name	Description	Reset
[3]	P3	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[2]	P2	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[1]	P1	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[0]	P0	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP0R1_EL1 is implemented only in implementations that support 6 or more bits of priority. ICV_AP0R2_EL1 and ICV_AP0R3_EL1 are implemented only in implementations that support 7 bits of priority. Unimplemented registers are **UNDEFINED**.

Writing to the active priority registers in any order other than the following order might result in **UNPREDICTABLE** behavior of the interrupt prioritization system:

- ICV_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 implemented only in implementations that support 6 or more bits of priority. ICV_AP0R2_EL1 and ICV_AP0R3_EL1 are implemented only in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order might result in UNPREDICTABLE behavior of the interrupt prioritization system:

- ICV_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.9.3 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-159: AArch64_icc_ap1r0_el1 bit assignments

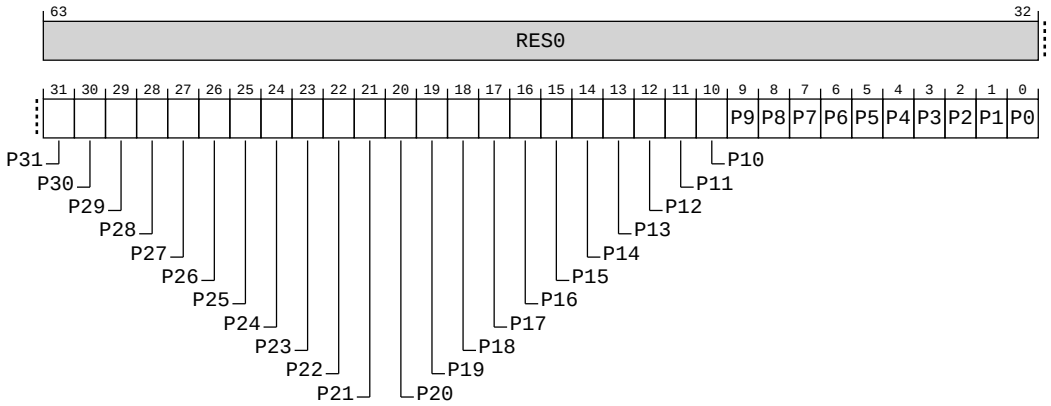


Table A-395: ICC_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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 implemented only in implementations that support 6 or more bits of priority. ICC_AP1R2_EL1 and ICC_AP1R3_EL1 are implemented only 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 implemented only in implementations that support 6 or more bits of priority. ICC_AP1R2_EL1 and ICC_AP1R3_EL1 are implemented only 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.9.4 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-160: AArch64_icv_ap1r0_el1 bit assignments

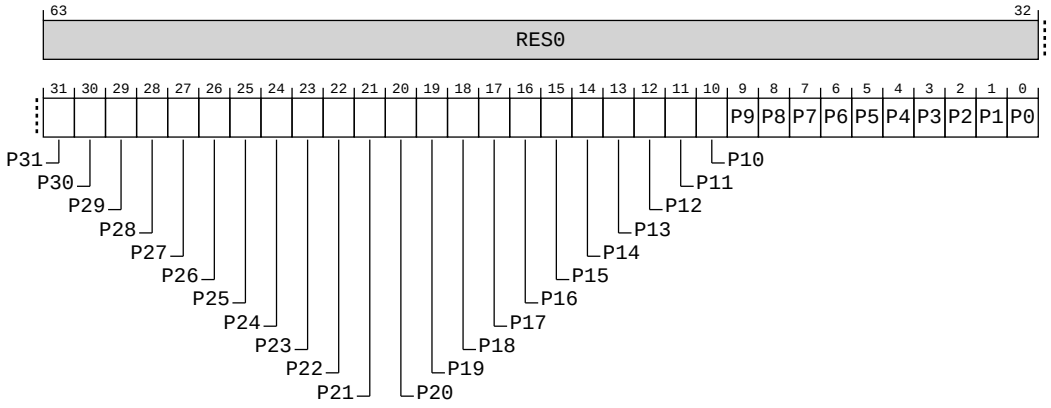


Table A-398: ICV_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[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

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. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[27]	P27	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[26]	P26	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[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

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. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[22]	P22	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[21]	P21	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[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

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. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[17]	P17	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[16]	P16	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[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

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. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[12]	P12	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[11]	P11	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[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

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. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[7]	P7	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[6]	P6	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[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

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. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[2]	P2	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[1]	P1	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	x
[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 implemented only in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are implemented only in implementations that support 7 bits of priority. Unimplemented registers are **UNDEFINED**.

Writing to the active priority registers in any order other than the following order might result in **UNPREDICTABLE** behavior of the interrupt prioritization system:

- 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 implemented only in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are implemented only in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order might result in UNPREDICTABLE behavior of the interrupt prioritization system:

- 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.9.5 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-161: AArch64_icc_ctlr_el1 bit assignments

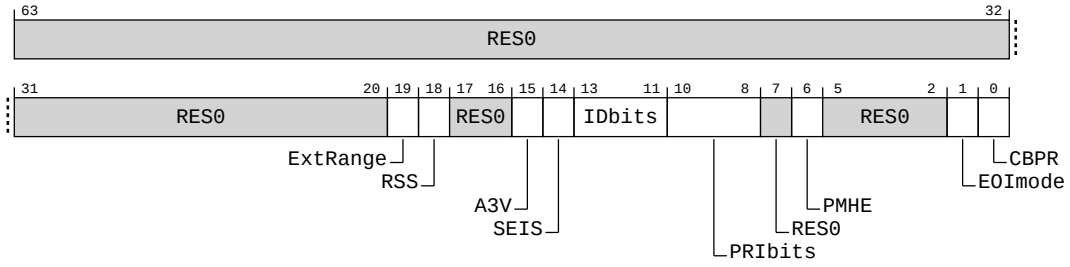


Table A-401: ICC_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x
[18]	RSS	Range Selector Support. Possible values are: **0b0** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	x
[17:16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **0b1** The CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: **0b0** The CPU interface logic does not support local generation of SEIs.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported: **0b000** 16 bits.	xxx
[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.	x
[5:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1]	EOImode	EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt: **0b0** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.	x
[0]	CBPR	Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts: **0b0** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts. **0b1** AArch64-ICC_BPR0_EL1 determines the preemption group for both Group 0 and Group 1 interrupts.	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.9.6 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-162: AArch64_icv_ctlr_el1 bit assignments

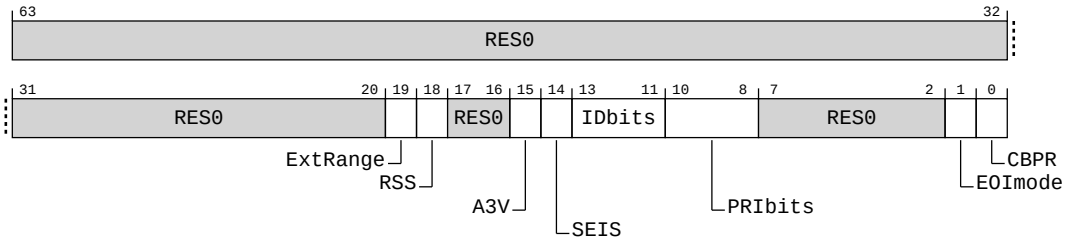


Table A-404: ICV_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x
[18]	RSS	Range Selector Support. Possible values are: **0b0** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	x
[17:16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **0b1** The virtual CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs: **0b0** The virtual CPU interface logic does not support local generation of SEIs.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported: **0b000** 16 bits.	xxx
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation must implement at least 32 levels of physical priority (5 priority bits). **Note:** This field always returns the number of priority bits implemented. The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPR0_EL1 and AArch64-ICV_BPR1_EL1. **0b100** 5 bits of priority are implemented	xxx
[7:2]	**RES0**	Reserved	**RES0**
[1]	EOImode	Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt: **0b0** AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICV_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.	x

Bits	Name	Description	Reset
[0]	CBPR	Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts: **0b0** AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts. **0b1** Non-secure reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Non-secure writes to AArch64-ICV_BPR1_EL1 are ignored. Secure reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1. Secure writes of AArch64-ICV_BPR1_EL1 modify AArch64-ICV_BPR0_EL1.	x

Access

MRS <Xt>, ICC_CTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

MSR ICC_CTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b100

Accessibility

MRS <Xt>, ICC_CTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if 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.9.7 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-163: AArch64_ich_ap0r0_el2 bit assignments

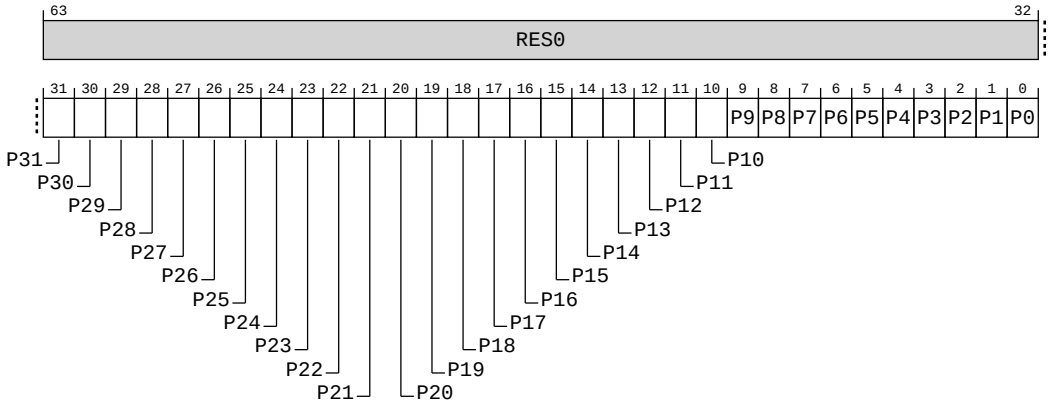


Table A-407: ICH_AP0R0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	P31	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[30]	P30	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[29]	P29	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[28]	P28	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[27]	P27	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[26]	P26	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[25]	P25	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

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.	0b0
[23]	P23	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[22]	P22	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[21]	P21	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[20]	P20	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[19]	P19	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[18]	P18	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

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.	0b0
[16]	P16	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[15]	P15	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[14]	P14	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[13]	P13	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[12]	P12	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[11]	P11	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

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.	0b0
[9]	P9	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[8]	P8	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[7]	P7	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[6]	P6	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[5]	P5	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[4]	P4	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

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.	0b0
[2]	P2	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[1]	P1	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0
[0]	P0	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0b0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop.	0b0

Software must ensure that ICH_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 implemented only in implementations that support 6 or more bits of preemption. ICH_AP0R2_EL2 and ICH_AP0R3_EL2 are implemented only in implementations that support 7 bits of preemption. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by 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 implemented only in implementations that support 6 or more bits of preemption. ICH_AP0R2_EL2 and ICH_AP0R3_EL2 are implemented only in implementations that support 7 bits of preemption. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by 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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x480 + (8 * 0)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_AP0R_EL2[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_AP0R_EL2[0];
```

MSR ICH_AP0R0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x480 + (8 * 0)] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_AP0R_EL2[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_AP0R_EL2[0] = X[t, 64];
```

A.9.8 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-164: AArch64_ich_ap1r0_el2 bit assignments

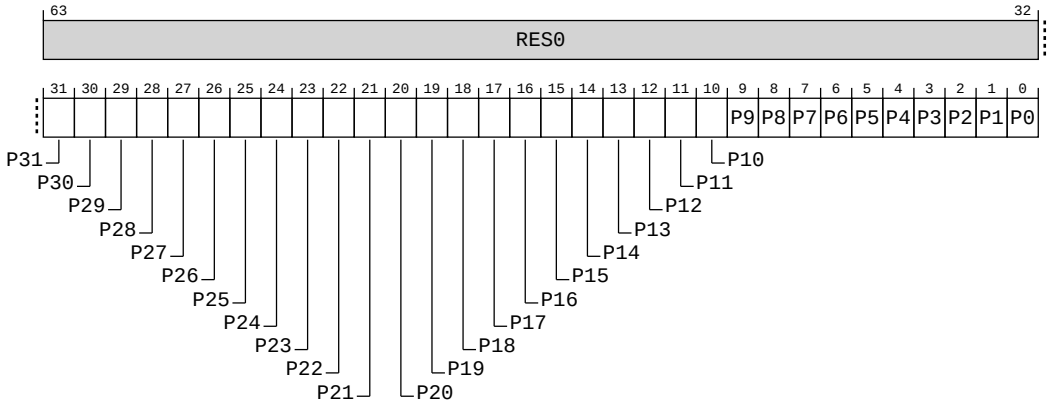


Table A-410: ICH_AP1R0_EL2 bit descriptions

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.	0b0
[30]	P30	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[29]	P29	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[28]	P28	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[27]	P27	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[26]	P26	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[25]	P25	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

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.	0b0
[23]	P23	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[22]	P22	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[21]	P21	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[20]	P20	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[19]	P19	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[18]	P18	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

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.	0b0
[16]	P16	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[15]	P15	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[14]	P14	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[13]	P13	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[12]	P12	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[11]	P11	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

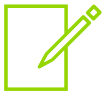
Bits	Name	Description	Reset
[10]	P10	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[9]	P9	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[8]	P8	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[7]	P7	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[6]	P6	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[5]	P5	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[4]	P4	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

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.	0b0
[2]	P2	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[1]	P1	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0
[0]	P0	Group 1 interrupt active priorities. Possible values of each bit are: **0b0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **0b1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop.	0b0

This register is always used for legacy VMs, regardless of the group of the virtual interrupt. Reads and writes to 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 implemented only in implementations that support 6 or more bits of preemption. ICH_AP1R2_EL2 and ICH_AP1R3_EL2 are implemented only in implementations that support 7 bits of preemption. Unimplemented registers are **UNDEFINED**.


Note

The number of bits of preemption is indicated by 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

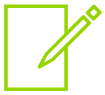
op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b000

MSR ICH_AP1R0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b000

Accessibility

ICH_AP1R1_EL2 is implemented only in implementations that support 6 or more bits of preemption. ICH_AP1R2_EL2 and ICH_AP1R3_EL2 are implemented only in implementations that support 7 bits of preemption. Unimplemented registers are UNDEFINED.


Note

The number of bits of preemption is indicated by 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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x4A0 + (8 * 0)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_AP1R_EL2[0];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_AP1R_EL2[0];
```

MSR ICH_AP1R0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x4A0 + (8 * 0)] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_AP1R_EL2[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_AP1R_EL2[0] = X[t, 64];
```

A.9.9 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

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0
xxxx	xxxx	xxxx	xxxx	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-165: AArch64_ich_vtr_el2 bit assignments

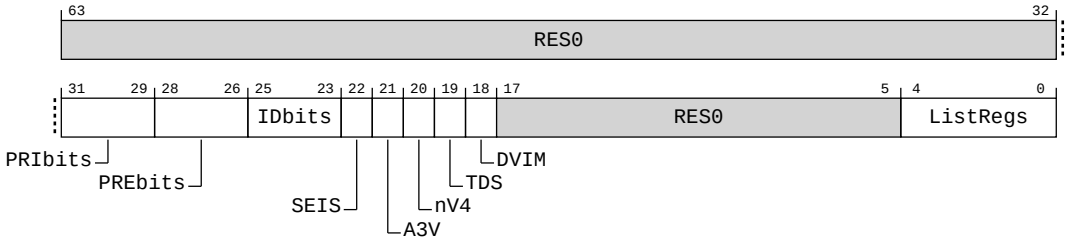


Table A-413: ICH_VTR_EL2 bit descriptions

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 nonzero 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. **0b1** Masking of Directly-injected Virtual Interrupts is supported.	x
[17:5]	**RES0**	Reserved	**RES0**
[4:0]	ListRegs	The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented. **0b00011** 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
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_VTR_EL2;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ICH_VTR_EL2;
```

A.9.10 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-166: AArch64_ich_lr0_el2 bit assignments

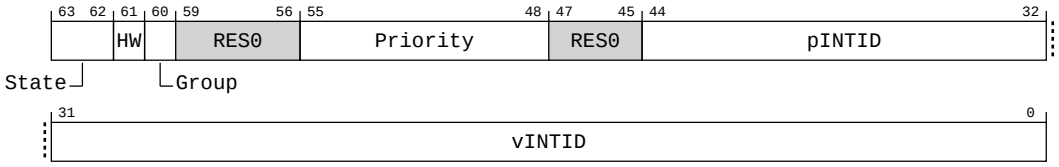


Table A-415: ICH_LR0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active.	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. When ICH_LR<n>_EL2.NMI is set to 1, this field is **RES0** and the virtual interrupt's priority is treated as `0x00`.	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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x400 + (8 * 0)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x400 + (8 * 0)] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_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.9.11 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-167: AArch64_ich_lr1_el2 bit assignments

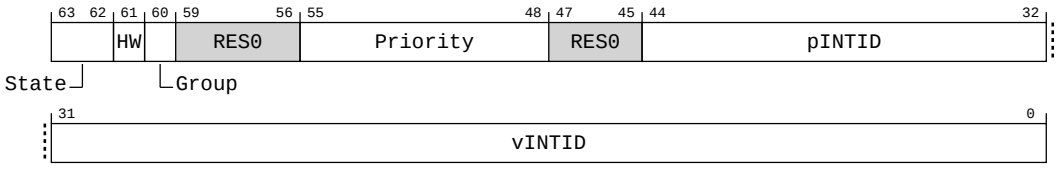


Table A-418: ICH_LR1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active.	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. When ICH_LR<n>_EL2.NMI is set to 1, this field is **RES0** and the virtual interrupt's priority is treated as `0x00`.	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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x400 + (8 * 1)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x400 + (8 * 1)] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_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.9.12 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-168: AArch64_ich_lr2_el2 bit assignments

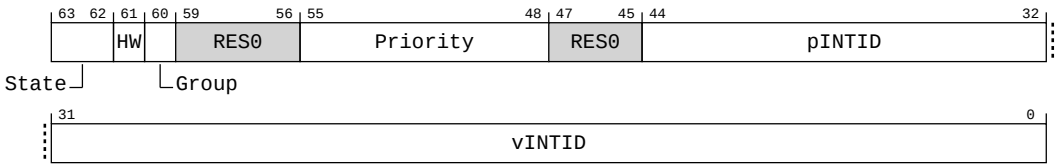


Table A-421: ICH_LR2_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active.	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. When ICH_LR<n>_EL2.NMI is set to 1, this field is **RES0** and the virtual interrupt's priority is treated as `0x00`.	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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x400 + (8 * 2)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x400 + (8 * 2)] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_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.9.13 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-169: AArch64_ich_lr3_el2 bit assignments

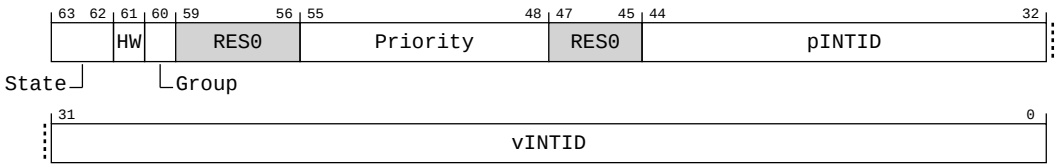


Table A-424: ICH_LR3_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	The state of the interrupt: **0b00** Invalid (Inactive). **0b01** Pending. **0b10** Active. **0b11** Pending and active.	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. When ICH_LR<n>_EL2.NMI is set to 1, this field is **RES0** and the virtual interrupt's priority is treated as `0x00`.	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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x400 + (8 * 3)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ICH_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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x400 + (8 * 3)] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_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.9.14 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 x0xx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-170: AArch64_icc_ctlr_el3 bit assignments

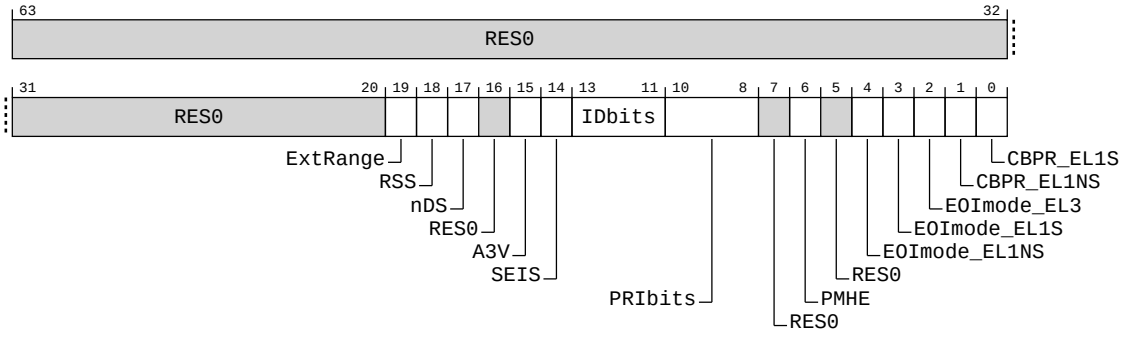


Table A-427: ICC_CTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:20]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[19]	ExtRange	Extended INTID range (read-only). **0b1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	x
[18]	RSS	Range Selector Support. **0b0** Targeted SGIs with affinity level 0 values of 0-15 are supported.	x
[17]	nDS	Disable Security not supported. Read-only and writes are ignored. **0b1** The CPU interface logic does not support disabling of security, and requires that security is not disabled.	x
[16]	**RES0**	Reserved	**RES0**
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. **0b1** The CPU interface logic supports nonzero values of the Aff3 field in SGI generation System registers.	x
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs: **0b0** The CPU interface logic does not support generation of SEIs.	x
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported. **0b000** 16 bits.	xxx
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of ext-GICD_CTLR.DS. The division between group priority and subpriority is defined in the binary point registers AArch64-ICC_BPR0_EL1 and AArch64-ICC_BPR1_EL1. This field determines the minimum value of ICC_BPR0_EL1. **0b100** 5 bits of priority are implemented	xxx
[7]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[6]	PMHE	Priority Mask Hint Enable. **0b0** Disables use of the priority mask register as a hint for interrupt distribution. **0b1** Enables use of the priority mask register as a hint for interrupt distribution.	0b0
[5]	**RES0**	Reserved	**RES0**
[4]	EOImode_EL1NS	EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0b0** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are **UNPREDICTABLE**. **0b1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.	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.	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.	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.	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.10 AArch64 Generic Timer registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-430: Generic Timer registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CNTKCTL_EL1	3	0	C14	C1	0	See individual bit resets.	64-bit	Counter-timer Kernel Control register
CNTFRQ_EL0	3	3	C14	C0	0	See individual bit resets.	64-bit	Counter-timer Frequency register
CNTPCT_EL0	3	3	C14	C0	1	See individual bit resets.	64-bit	Counter-timer Physical Count register
CNTVCT_EL0	3	3	C14	C0	2	See individual bit resets.	64-bit	Counter-timer Virtual Count register
CNTPCTSS_EL0	3	3	C14	C0	5	See individual bit resets.	64-bit	Counter-timer Self-Synchronized Physical Count register
CNTVCTSS_EL0	3	3	C14	C0	6	See individual bit resets.	64-bit	Counter-timer Self-Synchronized Virtual Count register
CNTP_TVAL_EL0	3	3	C14	C2	0	See individual bit resets.	64-bit	Counter-timer Physical Timer TimerValue register
CNTP_CTL_EL0	3	3	C14	C2	1	See individual bit resets.	64-bit	Counter-timer Physical Timer Control register
CNTP_CVAL_EL0	3	3	C14	C2	2	See individual bit resets.	64-bit	Counter-timer Physical Timer CompareValue register
CNTV_TVAL_EL0	3	3	C14	C3	0	See individual bit resets.	64-bit	Counter-timer Virtual Timer TimerValue register
CNTV_CTL_EL0	3	3	C14	C3	1	See individual bit resets.	64-bit	Counter-timer Virtual Timer Control register
CNTV_CVAL_EL0	3	3	C14	C3	2	See individual bit resets.	64-bit	Counter-timer Virtual Timer CompareValue register
CNTVOFF_EL2	3	4	C14	C0	3	See individual bit resets.	64-bit	Counter-timer Virtual Offset register
CNTPOFF_EL2	3	4	C14	C0	6	See individual bit resets.	64-bit	Counter-timer Physical Offset register
CNTHCTL_EL2	3	4	C14	C1	0	See individual bit resets.	64-bit	Counter-timer Hypervisor Control register
CNTHP_TVAL_EL2	3	4	C14	C2	0	See individual bit resets.	64-bit	Counter-timer Physical Timer TimerValue register (EL2)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CNTHP_CTL_EL2	3	4	C14	C2	1	See individual bit resets.	64-bit	Counter-timer Hypervisor Physical Timer Control register
CNTHP_CVAL_EL2	3	4	C14	C2	2	See individual bit resets.	64-bit	Counter-timer Physical Timer CompareValue register (EL2)
CNTHV_TVAL_EL2	3	4	C14	C3	0	See individual bit resets.	64-bit	Counter-timer Virtual Timer TimerValue Register (EL2)
CNTHV_CTL_EL2	3	4	C14	C3	1	See individual bit resets.	64-bit	Counter-timer Virtual Timer Control register (EL2)
CNTHV_CVAL_EL2	3	4	C14	C3	2	See individual bit resets.	64-bit	Counter-timer Virtual Timer CompareValue register (EL2)
CNTHVS_TVAL_EL2	3	4	C14	C4	0	See individual bit resets.	64-bit	Counter-timer Secure Virtual Timer TimerValue register (EL2)
CNTHVS_CTL_EL2	3	4	C14	C4	1	See individual bit resets.	64-bit	Counter-timer Secure Virtual Timer Control register (EL2)
CNTHVS_CVAL_EL2	3	4	C14	C4	2	See individual bit resets.	64-bit	Counter-timer Secure Virtual Timer CompareValue register (EL2)
CNTHPS_TVAL_EL2	3	4	C14	C5	0	See individual bit resets.	64-bit	Counter-timer Secure Physical Timer TimerValue register (EL2)
CNTHPS_CTL_EL2	3	4	C14	C5	1	See individual bit resets.	64-bit	Counter-timer Secure Physical Timer Control register (EL2)
CNTHPS_CVAL_EL2	3	4	C14	C5	2	See individual bit resets.	64-bit	Counter-timer Secure Physical Timer CompareValue register (EL2)
CNTPS_TVAL_EL1	3	7	C14	C2	0	See individual bit resets.	64-bit	Counter-timer Physical Secure Timer TimerValue register
CNTPS_CTL_EL1	3	7	C14	C2	1	See individual bit resets.	64-bit	Counter-timer Physical Secure Timer Control register
CNTPS_CVAL_EL1	3	7	C14	C2	2	See individual bit resets.	64-bit	Counter-timer Physical Secure Timer CompareValue register

A.11 AArch64 Other system control registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-431: Other system control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SCTLR_EL1	3	0	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL1)
CPACR_EL1	3	0	C1	C0	2	See individual bit resets.	64-bit	Architectural Feature Access Control Register
ZCR_EL1	3	0	C1	C2	0	See individual bit resets.	64-bit	SVE Control Register (EL1)
ACCDATA_EL1	3	0	C13	C0	5	See individual bit resets.	64-bit	Accelerator Data
SCTLR_EL2	3	4	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL2)
HCR_EL2	3	4	C1	C1	0	See individual bit resets.	64-bit	Hypervisor Configuration Register
CPTR_EL2	3	4	C1	C1	2	See individual bit resets.	64-bit	Architectural Feature Trap Register (EL2)
HSTR_EL2	3	4	C1	C1	3	See individual bit resets.	64-bit	Hypervisor System Trap Register
HFGRTR_EL2	3	4	C1	C1	4	See individual bit resets.	64-bit	Hypervisor Fine-Grained Read Trap Register
HFGWTR_EL2	3	4	C1	C1	5	See individual bit resets.	64-bit	Hypervisor Fine-Grained Write Trap Register
HFGITR_EL2	3	4	C1	C1	6	See individual bit resets.	64-bit	Hypervisor Fine-Grained Instruction Trap Register
ZCR_EL2	3	4	C1	C2	0	See individual bit resets.	64-bit	SVE Control Register (EL2)
HCRX_EL2	3	4	C1	C2	2	See individual bit resets.	64-bit	Extended Hypervisor Configuration Register
HDFGRTR_EL2	3	4	C3	C1	4	See individual bit resets.	64-bit	Hypervisor Debug Fine-Grained Read Trap Register
HDFGWTR_EL2	3	4	C3	C1	5	See individual bit resets.	64-bit	Hypervisor Debug Fine-Grained Write Trap Register
HAFGRTR_EL2	3	4	C3	C1	6	See individual bit resets.	64-bit	Hypervisor Activity Monitors Fine-Grained Read Trap Register
SCTLR_EL3	3	6	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL3)
ZCR_EL3	3	6	C1	C2	0	See individual bit resets.	64-bit	SVE Control Register (EL3)

A.11.1 ACCDATA_EL1, Accelerator Data

Holds the lower 32 bits of the data that is stored by an ST64BV0, Single-copy atomic 64-byte EL0 store instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Other system control registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-171: AArch64_accdata_el1 bit assignments

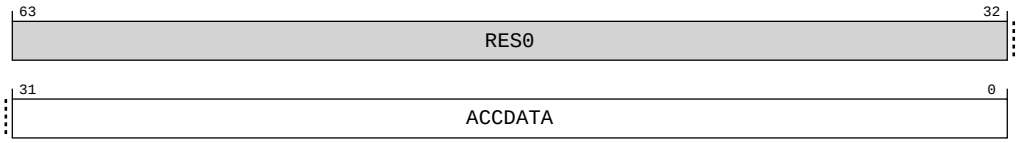


Table A-432: ACCDATA_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31:0]	ACCDATA	Accelerator Data field. Holds bits[31:0] of the data that is stored by an ST64BV0 instruction.	32{x}

Access

MRS <Xt>, ACCDATA_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b101

MSR ACCDATA_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b101

Accessibility

MRS <Xt>, ACCDATA_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.nACCDATA_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ACCDATA_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = ACCDATA_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ACCDATA_EL1;
```

MSR ACCDATA_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.nACCDATA_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ACCDATA_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.ADEn == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ACCDATA_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ACCDATA_EL1 = X[t, 64];
```

A.12 AArch64 RAS registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-435: RAS registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	0	C5	C3	0	See individual bit resets.	64-bit	Error Record ID Register
ERRSELR_EL1	3	0	C5	C3	1	See individual bit resets.	64-bit	Error Record Select Register
ERXFR_EL1	3	0	C5	C4	0	See individual bit resets.	64-bit	Selected Error Record Feature Register
ERXCTLR_EL1	3	0	C5	C4	1	See individual bit resets.	64-bit	Selected Error Record Control Register
ERXSTATUS_EL1	3	0	C5	C4	2	See individual bit resets.	64-bit	Selected Error Record Primary Status Register
ERXADDR_EL1	3	0	C5	C4	3	See individual bit resets.	64-bit	Selected Error Record Address Register
ERXPFGF_EL1	3	0	C5	C4	4	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Feature register
ERXPFGCTL_EL1	3	0	C5	C4	5	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Control register
ERXPFGCDN_EL1	3	0	C5	C4	6	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Countdown register
ERXMISC0_EL1	3	0	C5	C5	0	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	0	C5	C5	1	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	0	C5	C5	2	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	0	C5	C5	3	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 3
DISR_EL1	3	0	C12	C1	1	See individual bit resets.	64-bit	Deferred Interrupt Status Register
VSESR_EL2	3	4	C5	C2	3	See individual bit resets.	64-bit	Virtual SError Exception Syndrome Register
VDISR_EL2	3	4	C12	C1	1	See individual bit resets.	64-bit	Virtual Deferred Interrupt Status Register
MFAR_EL3	3	6	C6	C0	5	See individual bit resets.	64-bit	Physical Fault Address Register (EL3)

A.12.1 ERRIDR_EL1, Error Record ID Register

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-172: AArch64_erridr_el1 bit assignments

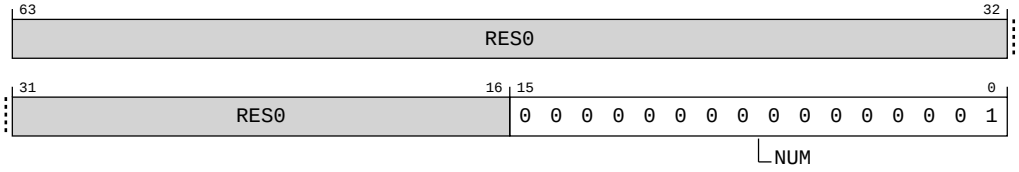


Table A-436: ERRIDR_EL1 bit descriptions

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. **0b0000000000000001** One Record Present.	0x0001

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERRIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.2 ERRSELR_EL1, Error Record Select Register

Selects an error record to be accessed through the Error Record System registers.

Configurations

If AArch64-ERRIDR_EL1 indicates that zero error records are implemented, then it is IMPLEMENTATION DEFINED whether ERRSELR_EL1 is UNDEFINED or RES0.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-173: AArch64_errselr_el1 bit assignments

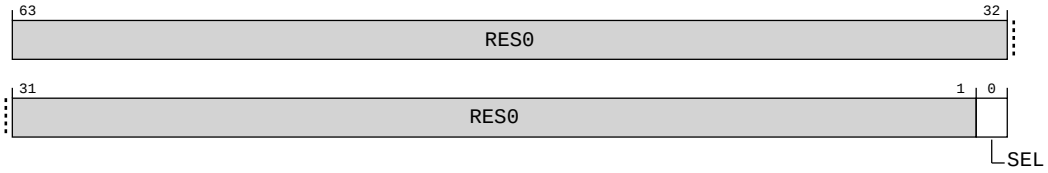


Table A-438: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	SEL	**0b0** Selects record 0, 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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t, 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.12.3 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 x101 0001 xxxx xxxx xxxx xxxx 1xxx xx00 0001 0000 1010 1001 1010 xx10
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-174: AArch64_erxfr_el1 bit assignments

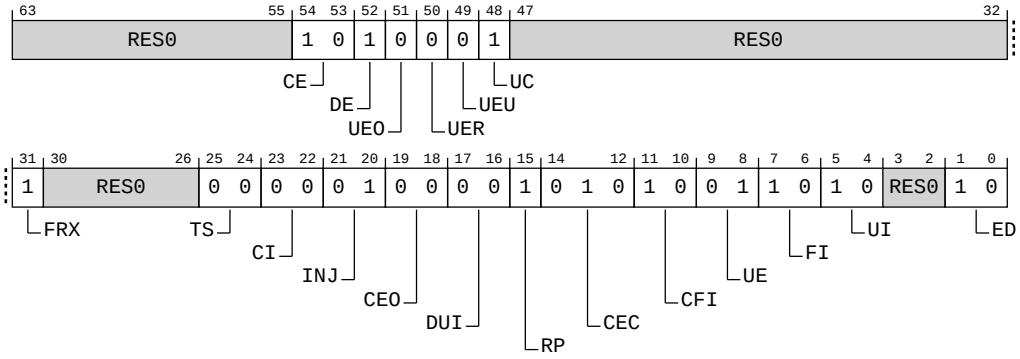


Table A-441: ERXFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:55]	RES0	Reserved	RES0
[54:53]	CE	Corrected Error recording. Describes the types of Corrected errors the node can record, if any. **0b10** Records only non-specific Corrected errors. That is, Corrected errors recorded by setting ERXSTATUS_EL1.CE to 0b10.	0b10
[52]	DE	Deferred Error recording. Describes whether the node supports recording Deferred errors. **0b1** Records Deferred errors.	0b1
[51]	UEO	Latent or Restartable Error recording. Describes whether the node supports recording Latent or Restartable errors. **0b0** Does not record Latent or Restartable errors.	0b0
[50]	UER	Signaled or Recoverable Error recording. Describes whether the node supports recording Signaled or Recoverable errors. **0b0** Does not record Signaled or Recoverable errors.	0b0
[49]	UEU	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b0** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b0
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b1** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b1

Bits	Name	Description	Reset
[47:32]	**RES0**	Reserved	**RES0**
[31]	FRX	Feature Register extension. Defines whether ERXFR_EL1[63:48] are architecturally defined. **0b1** ERXFR_EL1[63:48] are defined by the architecture.	0b1
[30:26]	**RES0**	Reserved	**RES0**
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, ERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. **0b00** The node does not support a timestamp register.	0b00
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. **0b00** Does not support the critical error interrupt. ERXCTLR_EL1.CI is **RES0**.	0b00
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. **0b01** The node implements the RAS Common Fault Injection Model Extension. See ERXPFGF_EL1 for more information.	0b01
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior when a second Corrected error is detected after a first Corrected error has been recorded by an error record <m> owned by the node. **0b00** Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then ERXSTATUS_EL1.OF is set to 0b1.	0b00
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the control for enabling error recovery interrupts on deferred errors are implemented. **0b00** Does not support the control for enabling error recovery interrupts on deferred errors. ERXCTLR_EL1.DUI is **RES0**.	0b00
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. **0b1** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	0b1
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in ERXMISC0_EL1 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERXMISC0_EL1[39:32].	0b010
[11:10]	CFI	Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented. **0b10** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI.	0b10

Bits	Name	Description	Reset
[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]	**RES0**	Reserved	**RES0**
[1:0]	ED	Error reporting and logging. Indicates whether error record <n> is the first record owned the node, and, if so, whether it implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ERXCTLR_EL1.ED.	0b10

Access

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.4 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-175: AArch64_erxctlr_el1 bit assignments

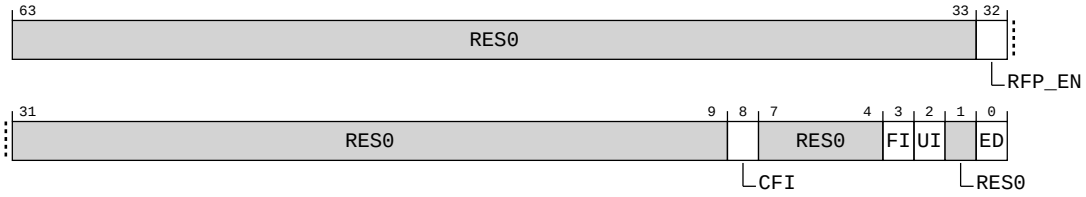


Table A-443: ERXCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:33]	**RES0**	Reserved	**RES0**
[32]	RFP_EN	Register File Parity error reporting and logging enable. When disabled, the node behaves as if Register File Parity error detection is disabled, and no RFP errors are recorded or signaled by the core. This control applies to Register File Parity errors when error logging and reporting is enabled via ext-ERR<n>CTRL.ED. **0b0** Register File Parity error reporting disabled. **0b1** Register File Parity error reporting enabled.	x
[31:9]	**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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.5 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-176: AArch64_erxstatus_el1 bit assignments

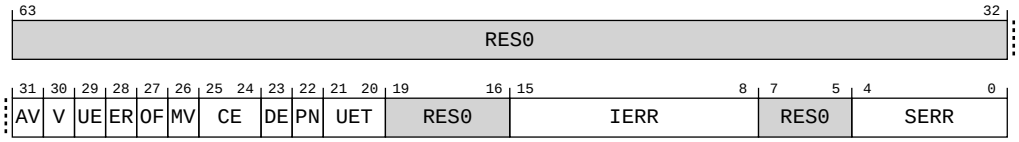


Table A-446: ERXSTATUS_EL1 bit descriptions

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
[19:16]	**RES0**	Reserved	**RES0**
[15:8]	IERR	Used with primary error code SERR value for additional information about the error. The core uses IERR values in conjunction with SERR code 0b11010 to provide the source of internal state parity error if identifiable. When other SERR codes are valid, IERR information may be stale or invalid and shall not be used. The possible values are: **0b00000000** Unknown source of internal state parity error **0b00000001** Parity error on General Register File **0b00000010** Parity error on Vector Register File	8{x}
[7:5]	**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. **0b11010** Parity error on internal state of the core including register files protected via Register File Parity 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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXSTATUS_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXSTATUS_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.6 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
```

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0	

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-177: AArch64_erxaddr_el1 bit assignments

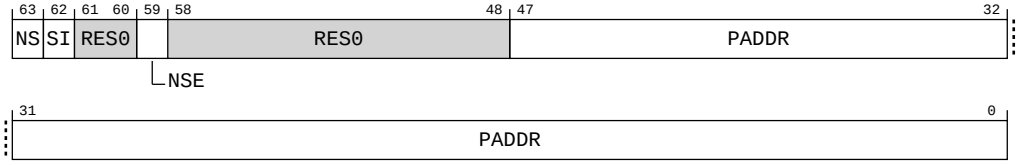


Table A-449: ERXADDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	**When IsFeatureImplemented(FEAT_RME)** Non-secure attribute. With ERR<n>ADDR.NSE, indicates the physical address space of the recorded location. **0b0** When ERR<n>ADDR.NSE == 0: ERR<n>ADDR.PADDR is a Secure address. When ERR<n>ADDR.NSE == 1: ERR<n>ADDR.PADDR is a Root address. **0b1** When ERR<n>ADDR.NSE == 0: ERR<n>ADDR.PADDR is a Non-secure address. When ERR<n>ADDR.NSE == 1: ERR<n>ADDR.PADDR is a Realm address. **When !IsFeatureImplemented(FEAT_RME)** Non-secure attribute. **0b0** ERR<n>ADDR.PADDR is a Secure address. **0b1** ERR<n>ADDR.PADDR is a Non-secure address. **Otherwise** RES0	x

Bits	Name	Description	Reset
[62]	SI	**When IsFeatureImplemented(FEAT_RME)** Secure Incorrect. Indicates whether ERR<n>ADDR.{NS, NSE} are valid. **0b0** ERR<n>ADDR.{NS, NSE} are correct. That is, they match the programmers' view of the physical address space for the recorded location. **0b1** ERR<n>ADDR.{NS, NSE} might not be correct, and might not match the programmers' view of the physical address space for the recorded location. **When !IsFeatureImplemented(FEAT_RME)** Secure Incorrect. Indicates whether ERR<n>ADDR.NS is valid. **0b0** ERR<n>ADDR.NS is correct. That is, it matches the programmers' view of the Non-secure attribute for the recorded location. **0b1** ERR<n>ADDR.NS might not be correct, and might not match the programmers' view of the Non-secure attribute for the recorded location. **Otherwise** RES0	x
[61:60]	**RES0**	Reserved	**RES0**
[59]	NSE	**When IsFeatureImplemented(FEAT_RME)** Physical Address Space. Together with ERR<n>ADDR.NS, indicates the address space for ERR<n>ADDR.PADDR. **Otherwise** RES0	x
[58:48]	**RES0**	Reserved	**RES0**
[47:0]	PADDR	Physical Address [47:0]. Address of the recorded location	48{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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.7 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 x111 xxxx xxxx xxxx xxx1 1010 0110 0011
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-178: AArch64_erxpfgf_el1 bit assignments

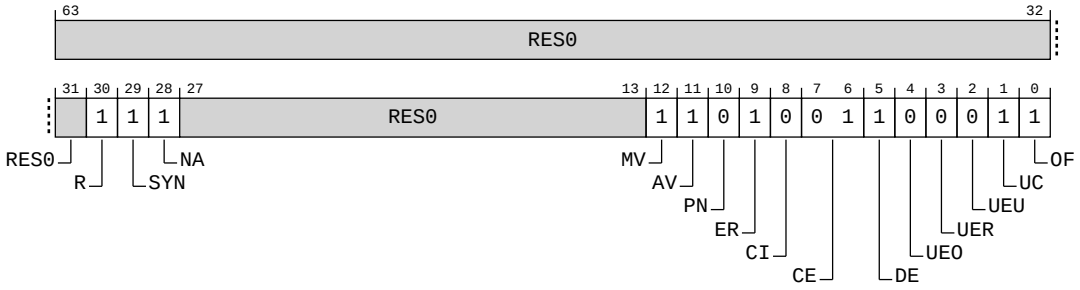


Table A-452: ERXPFGF_EL1 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	R	Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. The value is: **0b1** Error Generation Counter restart mode is implemented and is controlled by ERXPFGCTL_EL1.R. ERXPFGCTL_EL1.R is a read/write field	0b1
[29]	SYN	Syndrome. Fault syndrome injection. The value is: **0b1** When an injected error is recorded, the node does not update the ext-ERR<n>STATUS.{IERR, SERR} fields. ext-ERR<n>STATUS.{IERR, SERR} are writable when ext-ERR<n>STATUS.V is 0.	0b1
[28]	NA	No access required. Defines whether this component fakes detection of the error on an access to the component or spontaneously in the fault injection state. **0b1** The component fakes detection of the error spontaneously in the fault injection state.	0b1
[27:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. **0b1** When an injected error is recorded, the node might update some, but not all ERR<n>MISC<m> syndrome fields: • If any syndrome is recorded by the node in the ERR<n>MISC<m> registers, then ext-ERR<n>STATUS.MV is set to 1. • Otherwise, ext-ERR<n>STATUS.MV is set to ext-ERR<n>PFGCTL.MV. **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**.	0b1

Bits	Name	Description	Reset
[11]	AV	Address syndrome. Address syndrome injection. The value is: **0b1** When an injected error is recorded, the node does not update ext-ERR<n>ADDR and does: • Sets ext-ERR<n>STATUS.AV to 1.	0b1
[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: **0b1** When an injected error is recorded, ext-ERR<n>STATUS.ER is set to ext-ERR<n>PFGCTL.ER. This behavior replaces the architecture-defined rules for setting the ER bit.	0b1
[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
[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

Bits	Name	Description	Reset
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.OF status flag. The value is: **0b1** When an injected error is recorded, ext-ERR<n>STATUS.OF is set to ext-ERR<n>PFGCTL.OF. This behavior replaces the architecture-defined rules for setting the OF bit.	0b1

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGF_EL1 == '1' 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.12.8 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-179: AArch64_erxpfgctl_el1 bit assignments

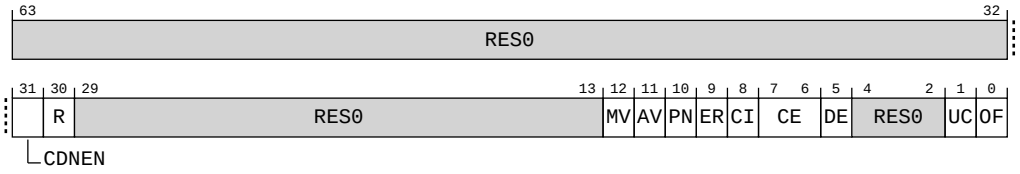


Table A-454: ERXPFGCTL_EL1 bit descriptions

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]	MV	Miscellaneous syndrome. The value written to AArch64-ERXSTATUS.MV when an injected error is recorded. **0b00** AArch64-ERXSTATUS.MV is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.MV is set to 1 when an injected error is recorded.	x
[11]	AV	Address syndrome. The value written to AArch64-ERXSTATUS.AV when an injected error is recorded. **0b00** AArch64-ERXSTATUS.AV is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.AV is set to 1 when an injected error is recorded.	x
[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]	ER	Error Reported flag. The value written to AArch64-ERXSTATUS.ER when an injected error is recorded. **0b00** AArch64-ERXSTATUS.ER is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.ER is set to 1 when an injected error is recorded.	x
[8]	CI	Critical Error flag. The value that is written to AArch64-ERXSTATUS.CI when an injected error is recorded. **0b0** AArch64-ERXSTATUS.CI is set to 0 when an injected error is recorded. **0b1** AArch64-ERXSTATUS.CI is set to 1 when an injected error is recorded.	0b0

Bits	Name	Description	Reset
[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are: **0b00** No error of this type will be generated. **0b01** A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10, might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b00
[5]	DE	Deferred Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[4:2]	**RES0**	Reserved	**RES0**
[1]	UC	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[0]	OF	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0

Access

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGCTL_EL1 == '1'
 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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFGCTL_EL1 == '1'
 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.12.9 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-180: AArch64_erxpfgcdn_el1 bit assignments

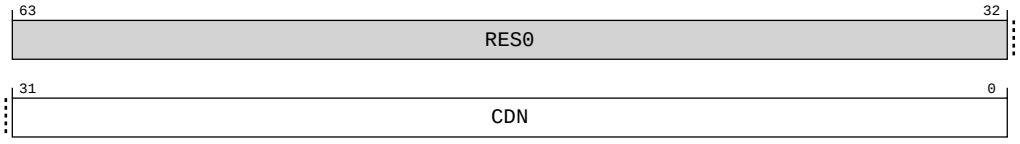


Table A-457: ERXPFGCDN_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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**.

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXPFGCDN_EL1 == '1'
 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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFGCDN_EL1 == '1'
 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.12.10 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-181: AArch64_erxmisc0_el1 bit assignments

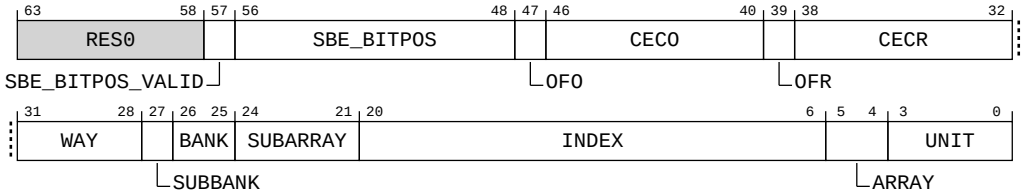


Table A-460: ERXMISC0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:58]	**RES0**	Reserved	**RES0**
[57]	SBE_BITPOS_VALID	Single Bit Error (SBE) bit position field ERXMISC0_EL1.SBE_BITPOS contains valid data **0b0** ERXMISC0_EL1.SBE_BITPOS does not contain valid data. **0b1** ERXMISC0_EL1.SBE_BITPOS contains valid data.	x
[56:48]	SBE_BITPOS	Single Bit Error (SBE) bit position. For a correctable error in a RAM with ECC (L1 data cache, L2 cache), indicates the bit position of the corrected error. Valid when ERXMISC0_EL1.SBE_BITPOS_VALID is 1'b1	9{x}
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.CECO is incremented and wraps through zero. **0b0** Other counter has not overflowed. **0b1** Other counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR. Unaffected by Cold or Warm reset.	7{x}

Bits	Name	Description	Reset
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.CECR is incremented and wraps through zero. **0b0** Repeat counter has not overflowed. **0b1** Repeat counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	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. Unaffected by Cold or Warm reset.	xxxx
[27]	SUBBANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Instruction Cache] • Indicates which subbank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Unaffected by Cold or Warm reset.	x

Bits	Name	Description	Reset
[26:25]	BANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 bank detected the error. 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
[24:21]	SUBARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 data 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
[20:6]	INDEX	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. [L1 Data Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size [L2 TLB] • Index of TLB RAM. Upper 4 bits are unused. [L1 Instruction Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. Unaffected by Cold or Warm reset.	15{x}

Bits	Name	Description	Reset
[5:4]	ARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] Indicates which array detected the error. The possible values are: • `0b00` L2 Tag RAM. • `0b01` L2 Data RAM. • `0b10` L2 TQ Data RAM. • `0b11` CHI Error. [L1 Data Cache] Indicates which array detected the error. The possible values are: • 00 LS Tag RAM 0. • 01 LS Tag RAM 1. • 10 LS Data RAM. • 11 LS Tag RAM 2. [L2 TLB] Indicates which array detected the error. The possible values are: • 00 Translation cache. • 01 GPT cache (when LEGACY_TZ_EN is 0). • 10 Reserved. • 11 Reserved. [L1 Instruction Cache] Indicates which array that detected the error, Data Array has higher priority. The possible values are: • `0b00` Tag. • `0b01` Data. Unaffected by Cold or Warm reset.	`xx`
[3:0]	UNIT	Indicates the unit which detected the error. The possible values are: **`0b0001`** L1 Instruction Cache. **`0b0010`** L2 TLB. **`0b0100`** L1 Data Cache. **`0b1000`** L2 Cache.	`xxxx`

Access

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.11 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
```

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-182: AArch64_erxmisc1_el1 bit assignments

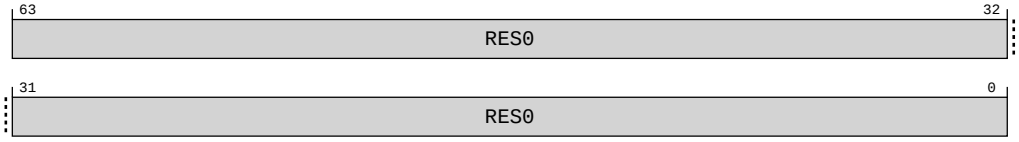


Table A-463: ERXMISC1_EL1 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.12 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-183: AArch64_erxmisc2_el1 bit assignments

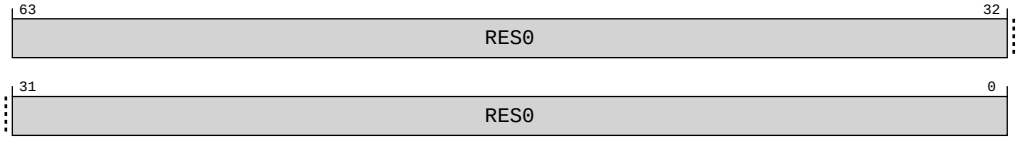


Table A-466: ERXMISC2_EL1 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.13 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
```

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-184: AArch64_erxmisc3_el1 bit assignments

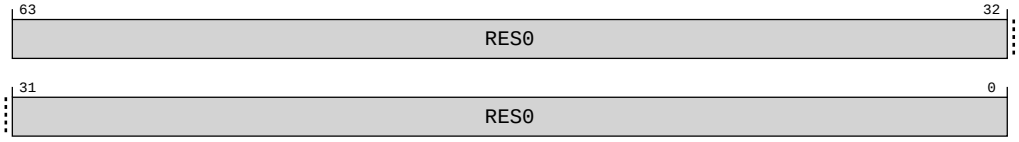


Table A-469: ERXMISC3_EL1 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.12.14 MFAR_EL3, Physical Fault Address Register (EL3)

Records the faulting physical address for a Granule Protection Check, synchronous External Abort, or SError exception taken to EL3.

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When IsFeatureImplemented(FEAT_RME) && the exception is a GPC exception

Figure A-185: AArch64_mfar_el3 bit assignments

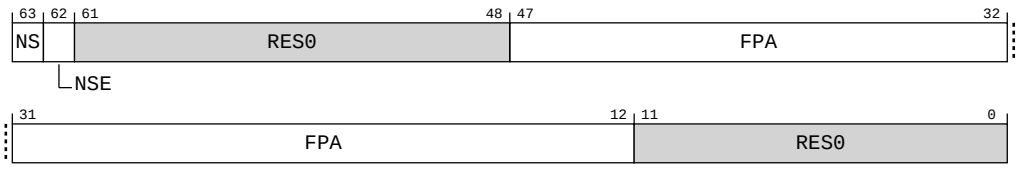


Table A-472: MFAR_EL3 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Together with MFAR_EL3.NSE, reports the physical address space of the access that triggered the exception. Table A-473: NSE description on page 697	x
[62]	NSE	Together with MFAR_EL3.NS, reports the physical address space of the access that triggered the exception. For a description of the values derived by evaluating NS and NSE together, see MFAR_EL3.NS.	x
[61:48]	**RES0**	Reserved	**RES0**
[47:12]	FPA	Bits [47:12] of the Faulting Physical Address.	36{x}
[11:0]	**RES0**	Reserved	**RES0**

Table A-473: NSE description

NSE	NS	Meaning
0b0	0b0	When Secure state is implemented, Secure. Otherwise reserved.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Access

MFAR_EL3 is not valid and reads **UNKNOWN** if AArch64-ESR_EL3.EC is recorded indicating an Abort or SError exception and AArch64-ESR_EL3.PFV is recorded as 0.

MRS <Xt>, MFAR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b101

MSR MFAR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0110	0b0000	0b101

Accessibility

MFAR_EL3 is not valid and reads UNKNOWN if AArch64-ESR_EL3.EC is recorded indicating an Abort or SError exception and AArch64-ESR_EL3.PFV is recorded as 0.
MRS <Xt>, MFAR_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] = MFAR_EL3;
```

MSR MFAR_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
    MFAR_EL3 = X[t, 64];
```

A.13 AArch64 Activity Monitors registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-476: Activity Monitors registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMCR_EL0	3	3	C13	C2	0	See individual bit resets.	64-bit	Activity Monitors Control Register
AMCFGR_EL0	3	3	C13	C2	1	See individual bit resets.	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	3	C13	C2	2	See individual bit resets.	64-bit	Activity Monitors Counter Group Configuration Register
AMUSERENR_EL0	3	3	C13	C2	3	See individual bit resets.	64-bit	Activity Monitors User Enable Register
AMCNTENCLR0_EL0	3	3	C13	C2	4	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 0
AMCNTENSET0_EL0	3	3	C13	C2	5	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 0
AMCNTENCLR1_EL0	3	3	C13	C3	0	See individual bit resets.	64-bit	Activity Monitors Count Enable Clear Register 1
AMCNTENSET1_EL0	3	3	C13	C3	1	See individual bit resets.	64-bit	Activity Monitors Count Enable Set Register 1
AMEVCNTR00_EL0	3	3	C13	C4	0	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR01_EL0	3	3	C13	C4	1	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0
AMEVCNTR02_EL0	3	3	C13	C4	2	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AMEVCNTR03_EL0	3	3	C13	C4	3	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 0
AMEVTYPER00_EL0	3	3	C13	C6	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	3	C13	C6	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	3	C13	C6	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	3	C13	C6	3	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVCNTR10_EL0	3	3	C13	C12	0	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR11_EL0	3	3	C13	C12	1	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 1
AMEVCNTR12_EL0	3	3	C13	C12	2	See individual bit resets.	64-bit	Activity Monitors Event Counter Registers 1
AMEVTYPER10_EL0	3	3	C13	C14	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	3	C13	C14	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	3	C13	C14	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1

A.13.1 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

```
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-186: AArch64_amcfgr_el0 bit assignments

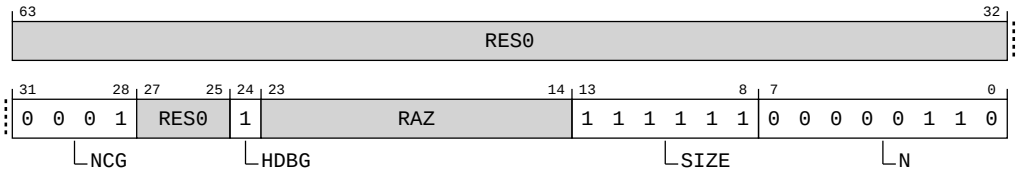


Table A-477: AMCFGR_EL0 bit descriptions

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.13.2 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-187: AArch64_amcgcr_el0 bit assignments

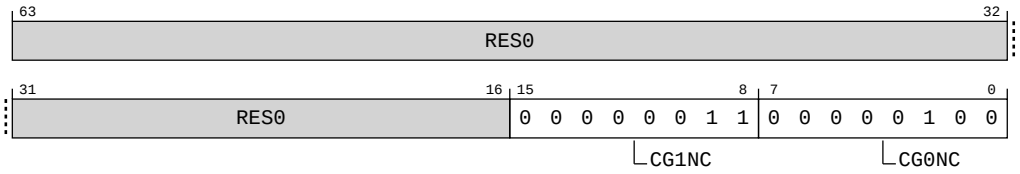


Table A-479: AMCGCR_EL0 bit descriptions

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.13.3 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-188: AArch64_amevcntr00_el0 bit assignments

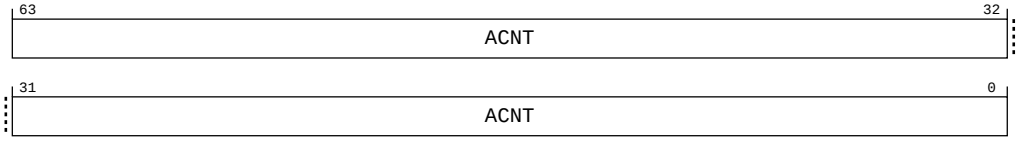


Table A-481: AMEVCNTR00_EL0 bit descriptions

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVCNTR00_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVCNTR00_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.4 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-189: AArch64_amevcntr01_el0 bit assignments

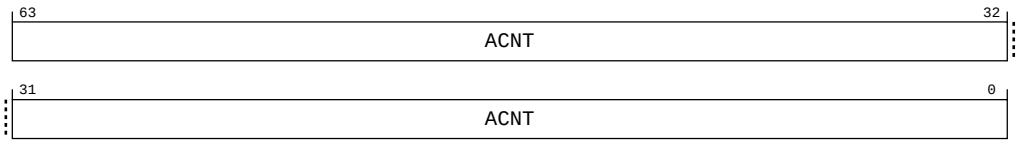


Table A-484: AMEVCNTR01_EL0 bit descriptions

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVCNTR01_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVCNTR01_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.5 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-190: AArch64_amevcntr02_el0 bit assignments

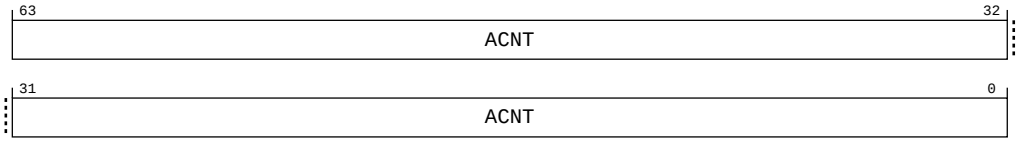


Table A-487: AMEVCNTR02_EL0 bit descriptions

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVCNTR02_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR0_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVCNTR02_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        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.13.6 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-191: AArch64_amevcntr03_el0 bit assignments

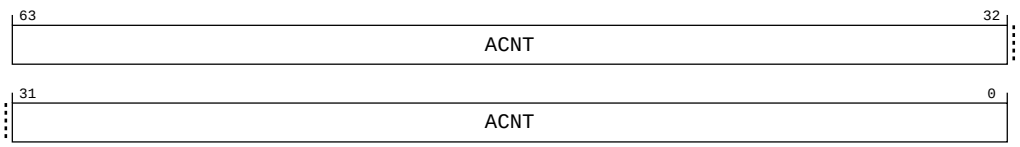


Table A-490: AMEVCNTR03_EL0 bit descriptions

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**.

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.

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVCNTR03_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        X[t, 64] = AMEVCNTR0_EL0[3];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVCNTR03_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.7 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-192: AArch64_amevtyper00_el0 bit assignments

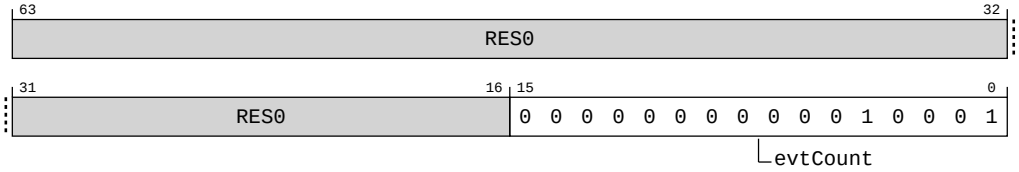


Table A-493: AMEVTYPER00_EL0 bit descriptions

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**.

Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER00_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b000

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.


Note

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER00_EL0

```
if PSTATE.EL == EL0 then
    if 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.13.8 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-193: AArch64_amevtyper01_el0 bit assignments

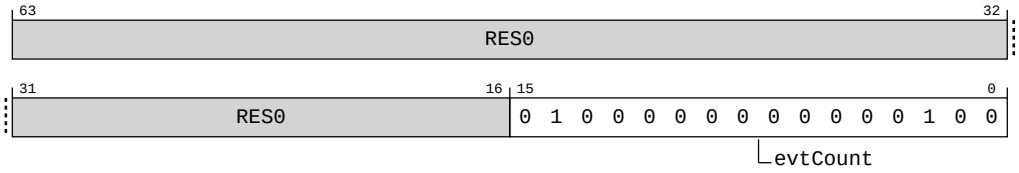


Table A-495: AMEVTYPER01_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-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.13.9 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-194: AArch64_amevtyper02_el0 bit assignments

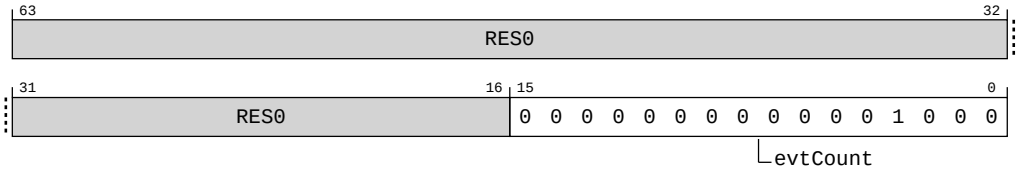


Table A-497: AMEVTYPER02_EL0 bit descriptions

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**.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER02_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b010

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER02_EL0

```
if PSTATE.EL == EL0 then
    if 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.13.10 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-195: AArch64_amevtyper03_el0 bit assignments

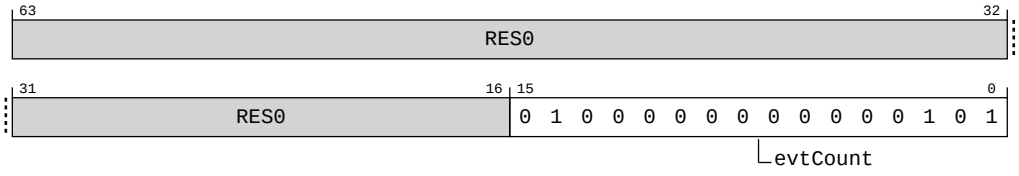


Table A-499: AMEVTYPER03_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter 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**.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER03_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0110	0b011

Accessibility

If <n> is greater than or equal to the number of architected activity monitor event counters, reads and writes of AMEVTYPER0<n>_EL0 are UNDEFINED.

AArch64-AMCGCR_EL0.CG0NC identifies the number of architected activity monitor event counters.

MRS <Xt>, AMEVTYPER03_EL0

```
if PSTATE.EL == EL0 then
    if 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.13.11 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-196: AArch64_amevcntr10_el0 bit assignments

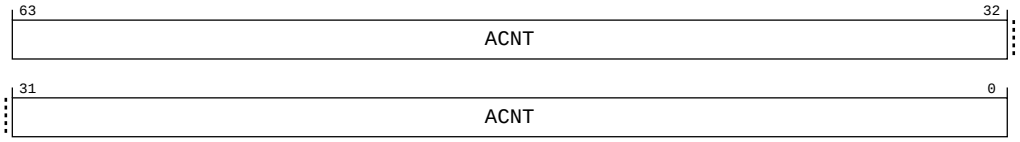


Table A-501: AMEVCNTR10_EL0 bit descriptions

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**.

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.

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVCNTR10_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[0];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVCNTR10_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.12 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-197: AArch64_amevcntr11_el0 bit assignments

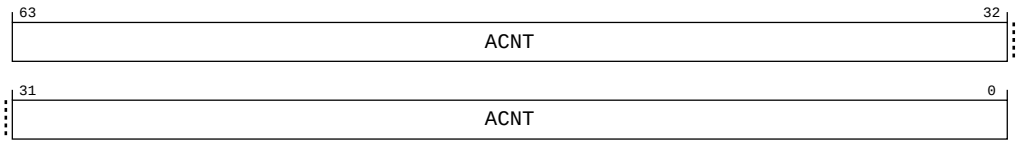


Table A-504: AMEVCNTR11_EL0 bit descriptions

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVCNTR11_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVCNTR11_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.13 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-198: AArch64_amevcntr12_el0 bit assignments

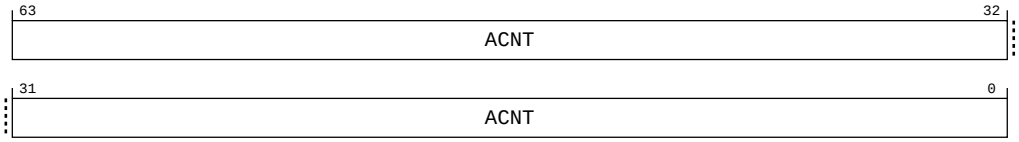


Table A-507: AMEVCNTR12_EL0 bit descriptions

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVCNTR12_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVCNTR1_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVCNTR12_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.14 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-199: AArch64_amevtyper10_el0 bit assignments

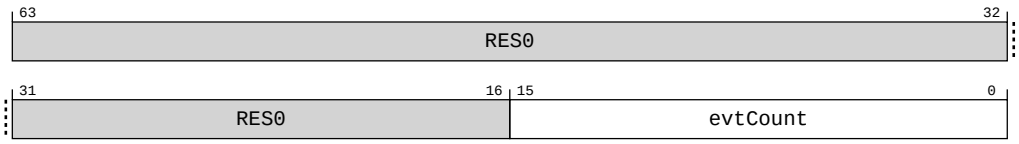


Table A-510: AMEVTYPER10_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:0]	evtCount	Event to count. The event number of the event that is counted by the 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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER10_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[0];
elsif PSTATE.EL == EL1 then
```

```
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER10_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.15 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-200: AArch64_amevtyper11_el0 bit assignments

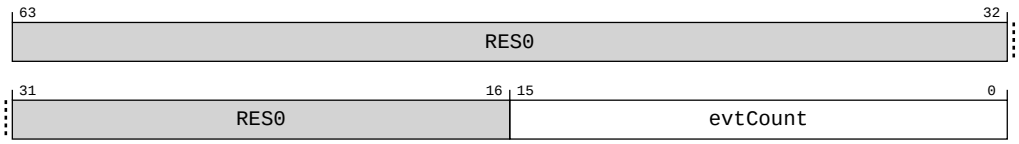


Table A-512: AMEVTYPER11_EL0 bit descriptions

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER11_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[1];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER11_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.13.16 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-201: AArch64_amevtyper12_el0 bit assignments

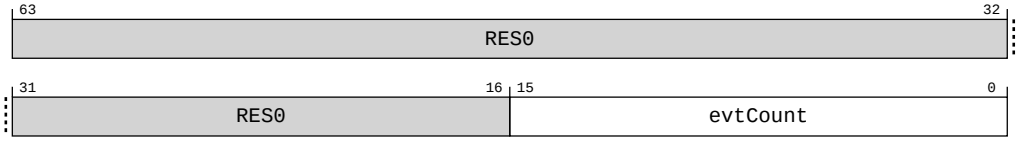


Table A-514: AMEVTYPER12_EL0 bit descriptions

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 EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' &&
 HAFGRTR_EL2.AMEVTYPER12_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = AMEVTYPER1_EL0[2];
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CPTR_EL2.TAM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HAFGRTR_EL2.AMEVTYPER12_EL0 == '1'
 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TAM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.14 AArch64 Branch Record Buffer Extension registers summary

The following summary table provides an overview of all Branch Record Buffer Extension registers in the core.

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

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

Table A-516: Branch Record Buffer Extension registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
BRBINF_n__EL1	2	1	C8	m[3:0]	m[4]:00	See individual bit resets.	64-bit	Branch Record Buffer Information Register <n>
BRBSRC_n__EL1	2	1	C8	m[3:0]	m[4]:01	See individual bit resets.	64-bit	Branch Record Buffer Source Address Register <n>
BRBTGT_n__EL1	2	1	C8	m[3:0]	m[4]:10	See individual bit resets.	64-bit	Branch Record Buffer Target Address Register <n>
BRBCR_EL1	2	1	C9	C0	0	See individual bit resets.	64-bit	Branch Record Buffer Control Register (EL1)
BRBCR_EL2	2	1	C9	C0	0	See individual bit resets.	64-bit	Branch Record Buffer Control Register (EL2)
BRBFCR_EL1	2	1	C9	C0	1	See individual bit resets.	64-bit	Branch Record Buffer Function Control Register
BRBTS_EL1	2	1	C9	C0	2	See individual bit resets.	64-bit	Branch Record Buffer Timestamp Register
BRBINFINJ_EL1	2	1	C9	C1	0	See individual bit resets.	64-bit	Branch Record Buffer Information Injection Register
BRBSRCINJ_EL1	2	1	C9	C1	1	See individual bit resets.	64-bit	Branch Record Buffer Source Address Injection Register
BRBTGTINJ_EL1	2	1	C9	C1	2	See individual bit resets.	64-bit	Branch Record Buffer Target Address Injection Register
BRBIDR0_EL1	2	1	C9	C2	0	See individual bit resets.	64-bit	Branch Record Buffer ID0 Register

A.14.1 BRBINF<n>_EL1, Branch Record Buffer Information Register <n> , n = 0 - 31

The information for Branch record n + (AArch64-BRBFCR_EL1.BANK × 32).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-202: AArch64_brbinf_n__el1 bit assignments

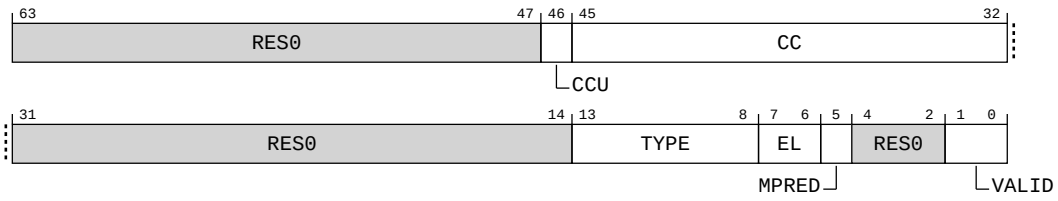


Table A-517: BRBINF<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:47]	RES0	Reserved	RES0
[46]	CCU	The number of PE clock cycles since the last Branch record entry is **UNKNOWN**. **0b0** Indicates that the number of PE clock cycles since the last Branch record is indicated by BRBINF<n>_EL1.CC. **0b1** Indicates that the number of PE clock cycles since the last Branch record is **UNKNOWN**. **When AArch64-BRBINF<n>_EL1.VALID == '00'** Access to this field is: **RES0** **Otherwise** Access to this field is: RO	x

Bits	Name	Description	Reset
[45:32]	CC	The number of PE clock cycles since the last Branch record entry. The format of this field uses a mantissa and exponent to express the cycle count value, as follows: • CC bits[7:0] indicate the mantissa M. • CC bits[13:8] indicate the exponent E. The cycle count is expressed using the following function: if IsZero(E) then UInt(M) else UInt('1':M:Zeros(UInt(E)-1)) If required, the cycle count is rounded to a multiple of $2^{(E-1)}$ towards zero before being encoded. A value of all ones in both the mantissa and exponent indicates the cycle count value exceeded the size of the cycle counter. **When AArch64-BRBINF<n>_EL1.CCU == '1' || AArch64-BRBINF<n>_EL1.VALID == '00'** Access to this field is: **RES0** **Otherwise** Access to this field is: RO	`14{x}`
[31:14]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[13:8]	TYPE	Branch type. **0b000000** Unconditional direct branch, excluding Branch with link. **0b000001** Indirect branch, excluding Branch with link, Return from subroutine, and Exception return. **0b000010** Direct Branch with link. **0b000011** Indirect Branch with link. **0b000101** Return from subroutine. **0b000111** Exception return. **0b001000** Conditional direct branch. **0b100001** Debug halt. **0b100010** Call. **0b100011** Trap. **0b100100** SError. **0b100110** Instruction debug. **0b100111** Data debug. **0b101010** Alignment. **0b101011** Inst Fault. **0b101100** Data Fault. **0b101110** IRQ. **0b101111** FIQ. **0b111001** Debug State Exit. **When AArch64-BRBINF<n>_EL1.VALID == '00'** Access to this field is: **RES0** **Otherwise** Access to this field is: RO	6{x}

Bits	Name	Description	Reset
[7:6]	EL	The Exception Level at the target address. **0b00** EL0. **0b01** EL1. **0b10** EL2. **When AArch64-BRBINF<n>_EL1.VALID == '00' || AArch64-BRBINF<n>_EL1.VALID == '10'** Access to this field is: **RES0** **Otherwise** Access to this field is: RO	xx
[5]	MPRED	Branch mispredict. **0b0** Branch was correctly predicted or the result of the prediction was not captured. **0b1** Branch was incorrectly predicted. **When AArch64-BRBINF<n>_EL1.VALID == '00' || AArch64-BRBINF<n>_EL1.VALID == '01' || AArch64-BRBINF<n>_EL1.TYPE[5] == '1'** Access to this field is: **RES0** **Otherwise** Access to this field is: RO	x
[4:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1:0]	VALID	The Branch record is valid. **0b00** This Branch record is not valid. The values of following fields are not valid: • AArch64-BRBTGT<n>_EL1.ADDRESS. • AArch64-BRBSRC<n>_EL1.ADDRESS. • BRBINF<n>_EL1.MPRED. • BRBINF<n>_EL1.LASTFAILED. • BRBINF<n>_EL1.T. • BRBINF<n>_EL1.EL. • BRBINF<n>_EL1.TYPE. • BRBINF<n>_EL1.CC. • BRBINF<n>_EL1.CCU. **0b01** This Branch record is valid. The values of following fields are not valid: • AArch64-BRBSRC<n>_EL1.ADDRESS. • BRBINF<n>_EL1.T. • BRBINF<n>_EL1.MPRED. **0b10** This Branch record is valid. The values of following fields are not valid: • AArch64-BRBTGT<n>_EL1.ADDRESS. • BRBINF<n>_EL1.EL. **0b11** This Branch record is valid.	xx

Access

BRBINF<n>_EL1 reads-as-zero if n + (AArch64-BRBFCR_EL1.BANK × 32) >= AArch64-BRBIDR0_EL1.NUMREC.

MRS <Xt>, BRBINF<m>_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1000	m[3:0]	m[4]:'00'

Accessibility

BRBINF<n>_EL1 reads-as-zero if n + (AArch64-BRBFCR_EL1.BANK × 32) >= AArch64-BRBIDR0_EL1.NUMREC.

MRS <Xt>, BRBINF<m>_EL1

```
integer m = UInt(op2<2>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBINF_EL1[m];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBINF_EL1[m];
elsif PSTATE.EL == EL3 then
    if m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBINF_EL1[m];
```

A.14.2 BRBSRC<n>_EL1, Branch Record Buffer Source Address Register <n> , n = 0 - 31

The source address of Branch record n + (AArch64-BRBFCR_EL1.BANK × 32).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-203: AArch64_brbsrc_n__el1 bit assignments

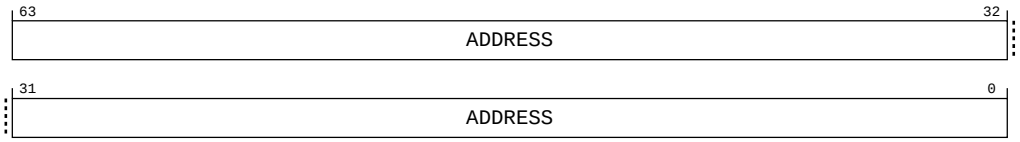


Table A-519: BRBSRC<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Source virtual address of the Branch record. When an indirect write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an **UNKNOWN** value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by DebugAddrTop(). The value in bits [P-1:0] are the value written. When an indirect write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value. **When AArch64-BRBINF<n>_EL1.VALID == '00' || AArch64-BRBINF<n>_EL1.VALID == '01'** Access to this field is: **RES0** **Otherwise** Access to this field is: RO	64{x}

Access

BRBSRC<n>_EL1 is **RES0** if n + (AArch64-BRBFCR_EL1.BANK × 32) >= AArch64-BRBIDR0_EL1.NUMREC.

MRS <Xt>, BRBSRC<m>_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1000	m[3:0]	m[4]:'01'

Accessibility

BRBSRC<n>_EL1 is RES0 if n + (AArch64-BRBFCR_EL1.BANK × 32) >= AArch64-BRBIDR0_EL1.NUMREC.
MRS <Xt>, BRBSRC<m>_EL1

```
integer m = UInt(op2<2>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBSRC_EL1[m];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBSRC_EL1[m];
elsif PSTATE.EL == EL3 then
    if m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBSRC_EL1[m];
```

A.14.3 BRBTGT<n>_EL1, Branch Record Buffer Target Address Register <n> , n = 0 - 31

The target address of Branch record n + (AArch64-BRBFCR_EL1.BANK × 32).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-204: AArch64_brbtgt_n__el1 bit assignments

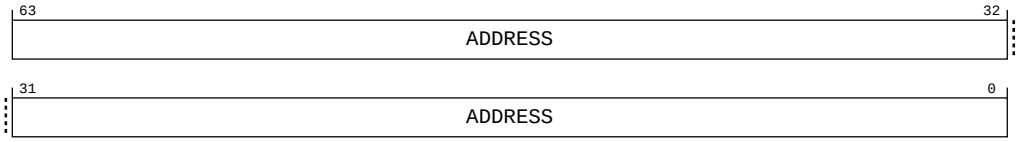


Table A-521: BRBTGT<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Target virtual address of the Branch record. When an indirect write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an **UNKNOWN** value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by DebugAddrTop(). The value in bits [P-1:0] are the value written. When an indirect write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value. **When AArch64-BRBINF<n>_EL1.VALID == '00' || AArch64-BRBINF<n>_EL1.VALID == '10'** Access to this field is: **RES0** **Otherwise** Access to this field is: RO	64{x}

Access

BRBTGT<n>_EL1 is **RES0** if n + (AArch64-BRBFCR_EL1.BANK × 32) >= AArch64-BRBIDR0_EL1.NUMREC.

MRS <Xt>, BRBTGT<m>_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1000	m[3:0]	m[4]:'10'

Accessibility

BRBTGT<n>_EL1 is RES0 if n + (AArch64-BRBFCR_EL1.BANK × 32) >= AArch64-BRBIDR0_EL1.NUMREC.
MRS <Xt>, BRBTGT<m>_EL1

```
integer m = UInt(op2<2>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBTGT_EL1[m];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBTGT_EL1[m];
elsif PSTATE.EL == EL3 then
    if m + (UInt(BRBFCR_EL1.BANK) * 32) >= NUM_BRBE_RECORDS then
        X[t, 64] = Zeros(64);
    else
        X[t, 64] = BRBTGT_EL1[m];
```

A.14.4 BRBCR_EL1, Branch Record Buffer Control Register (EL1)

Controls the Branch Record Buffer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-205: AArch64_brbcr_el1 bit assignments

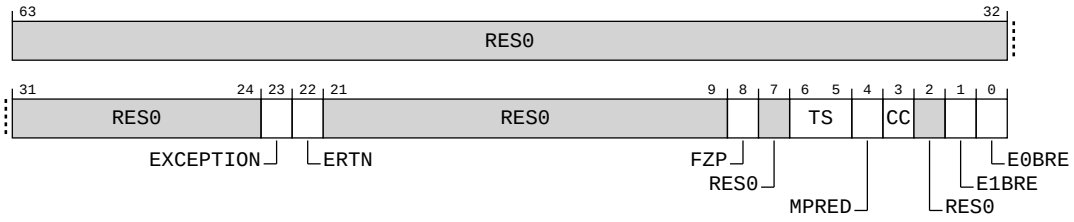


Table A-523: BRBCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**
[23]	EXCEPTION	Enable the recording of entry to EL1 via an exception. **0b0** Disable the recording of Branch records for exceptions when taken to EL1. **0b1** Enable the recording of Branch records for exceptions when taken to EL1.	x
[22]	ERTN	Allow the recording Branch records for exception return instructions from EL1. **0b0** Disable the recording Branch records for exception return instructions from EL1. **0b1** Enable the recording Branch records for exception return instructions from EL1.	x
[21:9]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[8]	FZP	Freeze BRBE on PMU overflow. **0b0** Branch recording is not affected by this control. **0b1** A BRBE freeze event occurs when a PMU overflow occurs.	x
[7]	**RES0**	Reserved	**RES0**
[6:5]	TS	Timestamp Control. **0b01** Virtual timestamp. The BRBE recorded timestamp is the physical counter value, minus the value of AArch64-CNTVOFF_EL2. **0b10** Guest physical timestamp. The BRBE recorded timestamp is the physical counter value minus a physical offset. If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of AArch64-CNTPOFF_EL2: • EL3 is implemented and AArch64-SCR_EL3.ECVEn == 0. • EL2 is implemented and AArch64-CNTHCTL_EL2.ECV == 0. **0b11** Physical timestamp. The BRBE recorded timestamp is the physical counter value.	xx
[4]	MPRED	Mask the recording of mispredicts. **0b0** Disable the recording of mispredict information. **0b1** Allow the recording of mispredict information.	x
[3]	CC	Enable the recording of cycle count information. **0b0** Disable the recording of cycle count information. **0b1** Allow the recording of cycle count information.	x
[2]	**RES0**	Reserved	**RES0**
[1]	E1BRE	EL1 Branch recording enable. **0b0** Branch recording prohibited at EL1. **0b1** Branch recording enabled at EL1.	0b0
[0]	E0BRE	EL0 Branch recording enable. **0b0** Branch recording prohibited at EL0. **0b1** Branch recording enabled at EL0.	0b0

Access

MRS <Xt>, BRBCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

MRS <Xt>, BRBCR_EL12

op0	op1	CRn	CRm	op2
0b10	0b101	0b1001	0b0000	0b000

MSR BRBCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

MSR BRBCR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b101	0b1001	0b0000	0b000

Accessibility

MRS <Xt>, BRBCR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x8E0];
    else
        X[t, 64] = BRBCR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = BRBCR_EL2;
    else
        X[t, 64] = BRBCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBCR_EL1;
```

MRS <Xt>, BRBCR_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        X[t, 64] = NVMem[0x8E0];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = BRBCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        X[t, 64] = BRBCR_EL1;
    else
        UNDEFINED;
```

MSR BRBCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x8E0] = X[t, 64];
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        BRBCR_EL2 = X[t, 64];
```

```
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBCR_EL1 = X[t, 64];
```

MSR BRBCR_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        NVMem[0x8E0] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            BRBCR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        BRBCR_EL1 = X[t, 64];
    else
        UNDEFINED;
```

A.14.5 BRBCR_EL2, Branch Record Buffer Control Register (EL2)

Controls the Branch Record Buffer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00
```

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-206: AArch64_brbcr_el2 bit assignments

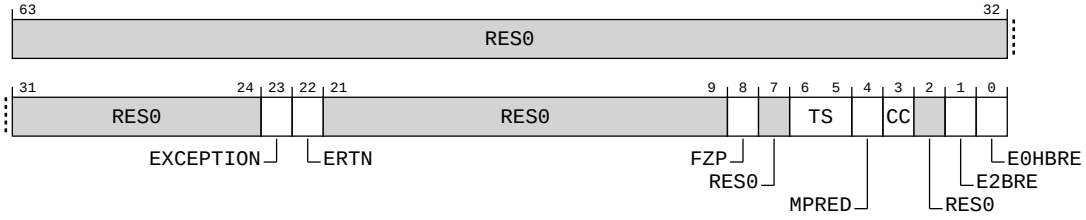


Table A-528: BRBCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:24]	**RES0**	Reserved	**RES0**
[23]	EXCEPTION	Enable the recording of entry to EL2 via an exception. **0b0** Disable the recording of Branch records for exceptions when taken to EL2. **0b1** Enable the recording of Branch records for exceptions when taken to EL2.	x
[22]	ERTN	Allow the recording Branch records for exception return instructions from EL2. **0b0** Disable the recording Branch records for exception return instructions from EL2. **0b1** Enable the recording Branch records for exception return instructions from EL2.	x
[21:9]	**RES0**	Reserved	**RES0**
[8]	FZP	Freeze BRBE on PMU overflow. **0b0** Branch recording is not affected by this control. **0b1** A BRBE freeze event occurs when a PMU overflow occurs.	x
[7]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[6:5]	TS	Timestamp Control. **0b00** Timestamp controlled by AArch64-BRBCR_EL1.TS. **0b01** Virtual timestamp. The BRBE recorded timestamp is the physical counter value, minus the value of AArch64-CNTVOFF_EL2. **0b10** Guest physical timestamp. The BRBE recorded timestamp is the physical counter value minus a physical offset. If any of the following are true, the physical offset is zero, otherwise the physical offset is the value of AArch64-CNTPOFF_EL2: • EL3 is implemented and AArch64-SCR_EL3.ECVEn == 0. • EL2 is implemented and AArch64-CNTHCTL_EL2.ECV == 0. **0b11** Physical timestamp. The BRBE recorded timestamp is the physical counter value.	xx
[4]	MPRED	Mask the recording of mispredicts. **0b0** Disable the recording of mispredict information. **0b1** Allow the recording of mispredict information.	x
[3]	CC	Enable the recording of cycle count information. **0b0** Disable the recording of cycle count information. **0b1** Allow the recording of cycle count information.	x
[2]	**RES0**	Reserved	**RES0**
[1]	E2BRE	EL2 Branch recording enable. **0b0** Branch recording prohibited at EL2. **0b1** Branch recording enabled at EL2.	0b0
[0]	E0HBRE	EL0 Branch recording enable. **0b0** Branch recording prohibited at EL0 when AArch64-HCR_EL2.TGE == 1. **0b1** Branch recording enabled at EL0 when AArch64-HCR_EL2.TGE == 1.	0b0

Access

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL2 using the register name BRBCR_EL2 or BRBCR_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

MRS <Xt>, BRBCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

MRS <Xt>, BRBCR_EL2

op0	op1	CRn	CRm	op2
0b10	0b100	0b1001	0b0000	0b000

MSR BRBCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b000

MSR BRBCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b100	0b1001	0b0000	0b000

Accessibility

When AArch64-HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL2 using the register name BRBCR_EL2 or BRBCR_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.
MRS <Xt>, BRBCR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        X[t, 64] = NVMem[0x8E0];
    else
        X[t, 64] = BRBCR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = BRBCR_EL2;
    else
        X[t, 64] = BRBCR_EL1;
```

```
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBCR_EL1;
```

MRS <Xt>, BRBCR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBCR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBCR_EL2;
```

MSR BRBCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x8E0] = X[t, 64];
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        BRBCR_EL2 = X[t, 64];
    else
        BRBCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
```

```
    BRBCR_EL1 = X[t, 64];
```

MSR BRBCR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBCR_EL2 = X[t, 64];
```

A.14.6 BRBFCR_EL1, Branch Record Buffer Function Control Register

Functional controls for the Branch Record Buffer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-207: AArch64_brbfcr_el1 bit assignments

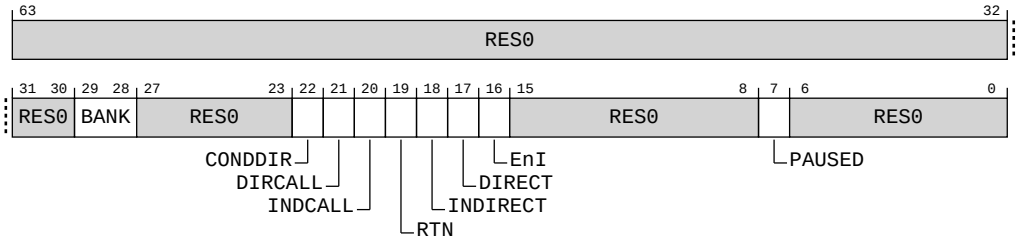


Table A-533: BRBFCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29:28]	BANK	Branch record buffer bank access control. **0b00** Select branch records 0 to 31. **0b01** Select branch records 32 to 63.	xx
[27:23]	**RES0**	Reserved	**RES0**
[22]	CONDDIR	Match on conditional direct branch instructions. **0b0** Do not match on conditional direct branch instructions. **0b1** Match on conditional direct branch instructions.	x
[21]	DIRCALL	Match on direct branch with link instructions. **0b0** Do not match on direct branch with link instructions. **0b1** Match on direct branch with link instructions.	x
[20]	INDCALL	Match on indirect branch with link instructions. **0b0** Do not match on indirect branch with link instructions. **0b1** Match on indirect branch with link instructions.	x
[19]	RTN	Match on function return instructions. **0b0** Do not match on function return instructions. **0b1** Match on function return instructions.	x

Bits	Name	Description	Reset
[18]	INDIRECT	Match on indirect branch instructions. **0b0** Do not match on indirect branch instructions. **0b1** Match on indirect branch instructions.	x
[17]	DIRECT	Match on unconditional direct branch instructions. **0b0** Do not match on unconditional direct branch instructions. **0b1** Match on unconditional direct branch instructions.	x
[16]	EnI	Include or exclude matches. **0b0** Include records for matches, and exclude records for non-matches. **0b1** Exclude records for matches, and include records for non-matches.	x
[15:8]	**RES0**	Reserved	**RES0**
[7]	PAUSED	Branch recording Paused status. **0b0** Branch recording is not Paused. **0b1** Branch recording is Paused.	x
[6:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, BRBFCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b001

MSR BRBFCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b001

Accessibility

MRS <Xt>, BRBFCR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBFCR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBFCR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBFCR_EL1;
```

MSR BRBFCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nBRBCTL == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBFCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBFCR_EL1 = X[t, 64];
```

A.14.7 BRBTS_EL1, Branch Record Buffer Timestamp Register

Captures the Timestamp value on a BRBE freeze event.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension 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-208: AArch64_brbts_el1 bit assignments

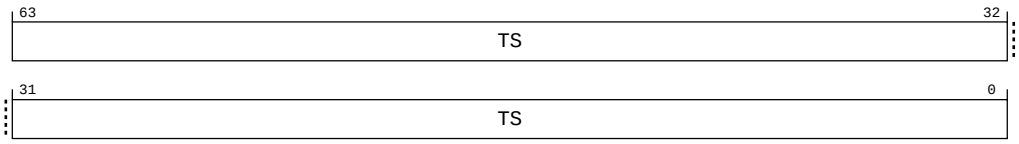


Table A-536: BRBTS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	TS	Timestamp value at the time of a BRBE freeze event.	0x0000000000000000

Access

MRS <Xt>, BRBTS_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b010

MSR BRBTS_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0000	0b010

Accessibility

MRS <Xt>, BRBTS_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTS_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTS_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBTS_EL1;
```

MSR BRBTS_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBTS_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        BRBTS_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBTS_EL1 = X[t, 64];
```

A.14.8 BRBINFINJ_EL1, Branch Record Buffer Information Injection Register

The information of a Branch record for injection.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

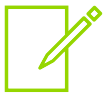
Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-209: AArch64_brbinfinj_el1 bit assignments

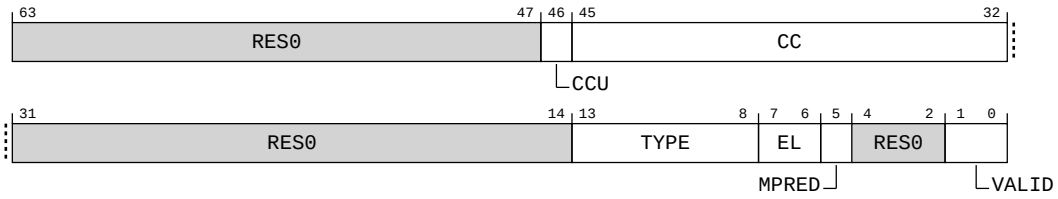


Table A-539: BRBINFINJ_EL1 bit descriptions

Bits	Name	Description	Reset
[63:47]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[46]	CCU	The number of PE clock cycles since the last Branch record entry is **UNKNOWN**. **0b0** Indicates that the number of PE clock cycles since the last Branch record is indicated by BRBINFINJ_EL1.CC. **0b1** Indicates that the number of PE clock cycles since the last Branch record is **UNKNOWN**. **When AArch64-BRBINFINJ_EL1.VALID == '00'** Access to this field is: **RES0** **Otherwise** Access to this field is: RW	x
[45:32]	CC	The number of PE clock cycles since the last Branch record entry. The format of this field uses a mantissa and exponent to express the cycle count value, as follows: • CC bits[7:0] indicate the mantissa M. • CC bits[13:8] indicate the exponent E. The cycle count is expressed using the following function: if IsZero(E) then UInt(M) else UInt('1':M:Zeros(UInt(E)-1)) If required, the cycle count is rounded to a multiple of $2^{(E-1)}$ towards zero before being encoded. A value of all ones in both the mantissa and exponent indicates the cycle count value exceeded the size of the cycle counter. **When AArch64-BRBINFINJ_EL1.CCU == '1' || AArch64-BRBINFINJ_EL1.VALID == '00'** Access to this field is: **RES0** **Otherwise** Access to this field is: RW	14{x}
[31:14]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[13:8]	TYPE	Branch type. **0b000000** Unconditional direct branch, excluding Branch with link. **0b000001** Indirect branch, excluding Branch with link, Return from subroutine, and Exception return. **0b000010** Direct Branch with link. **0b000011** Indirect Branch with link. **0b000101** Return from subroutine. **0b000111** Exception return. **0b001000** Conditional direct branch. **0b100001** Debug halt. **0b100010** Call. **0b100011** Trap. **0b100100** SError. **0b100110** Instruction debug. **0b100111** Data debug. **0b101010** Alignment. **0b101011** Inst Fault. **0b101100** Data Fault. **0b101110** IRQ. **0b101111** FIQ. **0b111001** Debug State Exit. **When AArch64-BRBINFINJ_EL1.VALID == '00'** Access to this field is: **RES0** **Otherwise** Access to this field is: RW	6{x}

Bits	Name	Description	Reset
[7:6]	EL	The Exception Level at the target address. **0b00** EL0. **0b01** EL1. **0b10** EL2. **When AArch64-BRBINFINJ_EL1.VALID == '00' || AArch64-BRBINFINJ_EL1.VALID == '10'** Access to this field is: **RES0** **Otherwise** Access to this field is: RW	xx
[5]	MPRED	Branch mispredict. **0b0** Branch was correctly predicted or the result of the prediction was not captured. **0b1** Branch was incorrectly predicted. **When AArch64-BRBINFINJ_EL1.VALID == '00' || AArch64-BRBINFINJ_EL1.VALID == '01' || AArch64-BRBINFINJ_EL1.TYPE[5] == '1'** Access to this field is: **RES0** **Otherwise** Access to this field is: RW	x
[4:2]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[1:0]	VALID	The Branch record is valid. **0b00** This Branch record is not valid. The values of following fields are not valid: • AArch64-BRBTGTINJ_EL1.ADDRESS. • AArch64-BRBSRCINJ_EL1.ADDRESS. • BRBINFINJ_EL1.MPRED. • BRBINFINJ_EL1.LASTFAILED. • BRBINFINJ_EL1.T. • BRBINFINJ_EL1.EL. • BRBINFINJ_EL1.TYPE. • BRBINFINJ_EL1.CC. • BRBINFINJ_EL1.CCU. **0b01** This Branch record is valid. The values of following fields are not valid: • AArch64-BRBSRCINJ_EL1.ADDRESS. • BRBINFINJ_EL1.T. • BRBINFINJ_EL1.MPRED. **0b10** This Branch record is valid. The values of following fields are not valid: • AArch64-BRBTGTINJ_EL1.ADDRESS. • BRBINFINJ_EL1.EL. **0b11** This Branch record is valid.	xx

Access

MRS <Xt>, BRBINFINJ_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b000

MSR BRBINFINJ_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b000

Accessibility

MRS <Xt>, BRBINFINJ_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBINFINJ_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBINFINJ_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBINFINJ_EL1;
```

MSR BRBINFINJ_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBINFINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    else
        BRBINFINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBINFINJ_EL1 = X[t, 64];
```

A.14.9 BRBSRCINJ_EL1, Branch Record Buffer Source Address Injection Register

The source address of a Branch record for injection.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-210: AArch64_brbsrcinj_el1 bit assignments

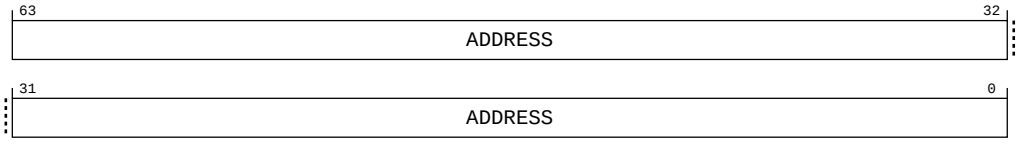


Table A-542: BRBSRCINJ_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Source virtual address of the Branch record. When a direct write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an **UNKNOWN** value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by DebugAddrTop(). The value in bits [P-1:0] are the value written. When a direct write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value. **When AArch64-BRBINFINJ_EL1.VALID == '00' || AArch64-BRBINFINJ_EL1.VALID == '01'** Access to this field is: **RES0** **Otherwise** Access to this field is: RW	64{x}

Access

MRS <Xt>, BRBSRCINJ_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b001

MSR BRBSRCINJ_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b001

Accessibility

MRS <Xt>, BRBSRCINJ_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBSRCINJ_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBSRCINJ_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBSRCINJ_EL1;
```

MSR BRBSRCINJ_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBSRCINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBSRCINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBSRCINJ_EL1 = X[t, 64];
```

A.14.10 BRBTGTINJ_EL1, Branch Record Buffer Target Address Injection Register

The target address of a Branch record for injection.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-211: AArch64_brbtgtinj_el1 bit assignments

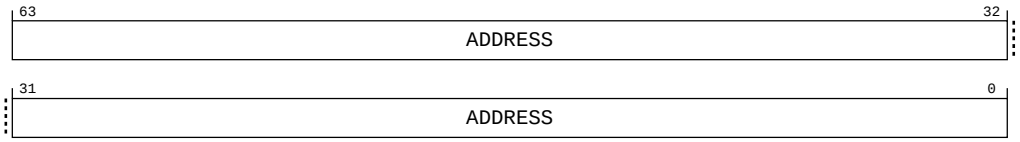


Table A-545: BRBTGTINJ_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	Target virtual address of the Branch record. When a direct write occurs with a value with ADDRESS bits [63:P] being other than all zeroes or all ones, an **UNKNOWN** value which is not all zeroes or all ones is written to bits [63:P]. P is defined as the virtual address size supported by the PE, as returned by DebugAddrTop(). The value in bits [P-1:0] are the value written. When a direct write occurs with a value with ADDRESS bits [63:P] being all zeroes or all ones, the written value is written to bits [63:0], and a read of the register returns the written value. **When AArch64-BRBINFINJ_EL1.VALID == '00' || AArch64-BRBINFINJ_EL1.VALID == '10'** Access to this field is: **RES0** **Otherwise** Access to this field is: RW	64{x}

Access

MRS <Xt>, BRBTGTINJ_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b010

MSR BRBTGTINJ_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	0b010

Accessibility

MRS <Xt>, BRBTGTINJ_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTGTINJ_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBTGTINJ_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBTGTINJ_EL1;
```

MSR BRBTGTINJ_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nBRBDATA == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBTGTINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
```

```
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        BRBTGTINJ_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    BRBTGTINJ_EL1 = X[t, 64];
```

A.14.11 BRBIDR0_EL1, Branch Record Buffer ID0 Register

Indicates the features of the branch buffer unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Branch Record Buffer Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-212: AArch64_brbidr0_el1 bit assignments

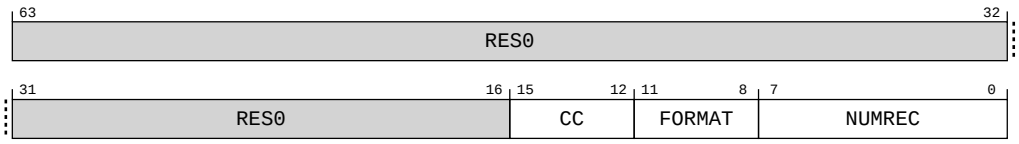


Table A-548: BRBIDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	**RES0**	Reserved	**RES0**
[15:12]	CC	Cycle counter support. Defined values are: **0b0101** 20-bit cycle counter implemented.	xxxx
[11:8]	FORMAT	Data format of records of the Branch record buffer. Defined values are: **0b0000** Format 0.	xxxx
[7:0]	NUMREC	Number of records supported. Defined values are: **0b00001000** 8 branch records implemented. **0b00010000** 16 branch records implemented. **0b00100000** 32 branch records implemented. **0b01000000** 64 branch records implemented.	8{x}

Access

MRS <Xt>, BRBIDR0_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0010	0b000

Accessibility

MRS <Xt>, BRBIDR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nBRBIDR == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBIDR0_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.SBRBE != '11' && SCR_EL3.NS == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.SBRBE == 'x0' && SCR_EL3.NS == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = BRBIDR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = BRBIDR0_EL1;
```

A.15 AArch64 Trace unit registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-550: Trace unit registers summary

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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCSEQEVR2	2	1	C0	C2	4	See individual bit resets.	64-bit	Sequencer State Transition Control Register <n>
TRCIDR10	2	1	C0	C2	6	See individual bit resets.	64-bit	ID Register 10
TRCSTATR	2	1	C0	C3	0	See individual bit resets.	64-bit	Trace Status Register
TRCIDR11	2	1	C0	C3	6	See individual bit resets.	64-bit	ID Register 11
TRCCONFIGR	2	1	C0	C4	0	See individual bit resets.	64-bit	Trace Configuration Register
TRCCNTCTLR0	2	1	C0	C4	5	See individual bit resets.	64-bit	Counter Control Register <n>
TRCIDR12	2	1	C0	C4	6	See individual bit resets.	64-bit	ID Register 12
TRCCNTCTLR1	2	1	C0	C5	5	See individual bit resets.	64-bit	Counter Control Register <n>
TRCIDR13	2	1	C0	C5	6	See individual bit resets.	64-bit	ID Register 13
TRCAUXCTLR	2	1	C0	C6	0	See individual bit resets.	64-bit	Auxiliary Control Register
TRCSEQRSTEVR	2	1	C0	C6	4	See individual bit resets.	64-bit	Sequencer Reset Control Register
TRCSEQSTR	2	1	C0	C7	4	See individual bit resets.	64-bit	Sequencer State Register
TRCEVENTCTL0R	2	1	C0	C8	0	See individual bit resets.	64-bit	Event Control 0 Register
TRCEXTINSELR0	2	1	C0	C8	4	See individual bit resets.	64-bit	External Input Select Register <n>
TRCCNTVR0	2	1	C0	C8	5	See individual bit resets.	64-bit	Counter Value Register <n>
TRCIDR0	2	1	C0	C8	7	See individual bit resets.	64-bit	ID Register 0
TRCEVENTCTL1R	2	1	C0	C9	0	See individual bit resets.	64-bit	Event Control 1 Register
TRCEXTINSELR1	2	1	C0	C9	4	See individual bit resets.	64-bit	External Input Select Register <n>
TRCCNTVR1	2	1	C0	C9	5	See individual bit resets.	64-bit	Counter Value Register <n>
TRCIDR1	2	1	C0	C9	7	See individual bit resets.	64-bit	ID Register 1
TRCRSR	2	1	C0	C10	0	See individual bit resets.	64-bit	Resources Status Register
TRCEXTINSELR2	2	1	C0	C10	4	See individual bit resets.	64-bit	External Input Select Register <n>
TRCIDR2	2	1	C0	C10	7	See individual bit resets.	64-bit	ID Register 2

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCEXTINSELR3	2	1	C0	C11	4	See individual bit resets.	64-bit	External Input Select Register <n>
TRCIDR3	2	1	C0	C11	7	See individual bit resets.	64-bit	ID Register 3
TRCTSCTLR	2	1	C0	C12	0	See individual bit resets.	64-bit	Timestamp Control Register
TRCIDR4	2	1	C0	C12	7	See individual bit resets.	64-bit	ID Register 4
TRCSYNCPR	2	1	C0	C13	0	See individual bit resets.	64-bit	Synchronization Period Register
TRCIDR5	2	1	C0	C13	7	See individual bit resets.	64-bit	ID Register 5
TRCCCCTLR	2	1	C0	C14	0	See individual bit resets.	64-bit	Cycle Count Control Register
TRCIDR6	2	1	C0	C14	7	See individual bit resets.	64-bit	ID Register 6
TRCBBCTLR	2	1	C0	C15	0	See individual bit resets.	64-bit	Branch Broadcast Control Register
TRCIDR7	2	1	C0	C15	7	See individual bit resets.	64-bit	ID Register 7
TRCSSCCR0	2	1	C1	C0	2	See individual bit resets.	64-bit	Single-shot Comparator Control Register <n>
TRCOSLSR	2	1	C1	C1	4	See individual bit resets.	64-bit	Trace OS Lock Status Register
TRCRSCTLR2	2	1	C1	C2	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR3	2	1	C1	C3	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR4	2	1	C1	C4	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR5	2	1	C1	C5	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR6	2	1	C1	C6	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR7	2	1	C1	C7	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR8	2	1	C1	C8	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCSSCSR0	2	1	C1	C8	2	See individual bit resets.	64-bit	Single-shot Comparator Control Status Register <n>
TRCRSCTLR9	2	1	C1	C9	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR10	2	1	C1	C10	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR11	2	1	C1	C11	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCRSCTLR12	2	1	C1	C12	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR13	2	1	C1	C13	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR14	2	1	C1	C14	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCRSCTLR15	2	1	C1	C15	0	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCACVR0	2	1	C2	C0	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR0	2	1	C2	C0	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR1	2	1	C2	C2	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR1	2	1	C2	C2	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR2	2	1	C2	C4	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR2	2	1	C2	C4	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR3	2	1	C2	C6	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR3	2	1	C2	C6	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR4	2	1	C2	C8	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR4	2	1	C2	C8	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR5	2	1	C2	C10	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR5	2	1	C2	C10	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR6	2	1	C2	C12	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR6	2	1	C2	C12	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCACVR7	2	1	C2	C14	0	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR7	2	1	C2	C14	2	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
TRCCIDCVR0	2	1	C3	C0	0	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>
TRCVMIDCVR0	2	1	C3	C0	1	See individual bit resets.	64-bit	Virtual Context Identifier Comparator Value Register <n>
TRCCIDCCTLR0	2	1	C3	C0	2	See individual bit resets.	64-bit	Context Identifier Comparator Control Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCVMIDCCTLR0	2	1	C3	C2	2	See individual bit resets.	64-bit	Virtual Context Identifier Comparator Control Register 0
TRCDEVID	2	1	C7	C2	7	See individual bit resets.	64-bit	Device Configuration Register
TRCCLAIMSET	2	1	C7	C8	6	See individual bit resets.	64-bit	Claim Tag Set Register
TRCCLAIMCLR	2	1	C7	C9	6	See individual bit resets.	64-bit	Claim Tag Clear Register
TRCAUTHSTATUS	2	1	C7	C14	6	See individual bit resets.	64-bit	Authentication Status Register
TRCDEVARCH	2	1	C7	C15	6	See individual bit resets.	64-bit	Device Architecture Register

A.15.1 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-213: AArch64_trcseqevr0 bit assignments

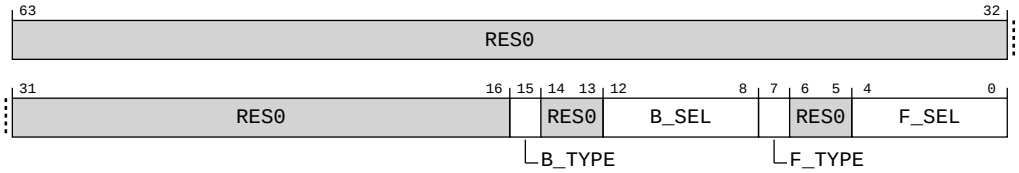


Table A-551: TRCSEQEVR0 bit descriptions

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.	`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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.2 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-214: AArch64_trccntrldvr0 bit assignments

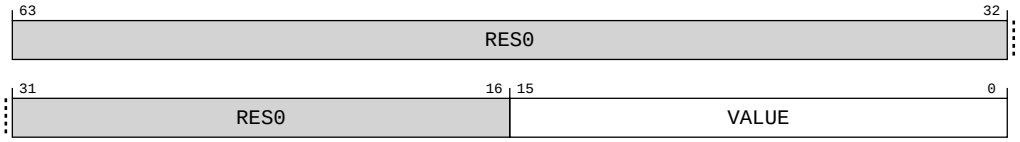


Table A-554: TRCCNTRLDVR0 bit descriptions

Bits	Name	Description	Reset
[63:16]	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.

MRS <Xt>, TRCCNTRLDVR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b101

MSR TRCCNTRLDVR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	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>, TRCCNTRLDVR0

```
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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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] = TRCCNTRLDVR[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] = TRCCNTRLDVR[0];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTRLDVR[0];
```

MSR TRCCNTRLDVR0, <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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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
        TRCCNTRLDVR[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
        TRCCNTRLDVR[0] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTRLDVR[0] = X[t, 64];
```

A.15.3 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-215: AArch64_trcidr8 bit assignments

Table A-557: 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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.4 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

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-216: AArch64_trcimspec0 bit assignments

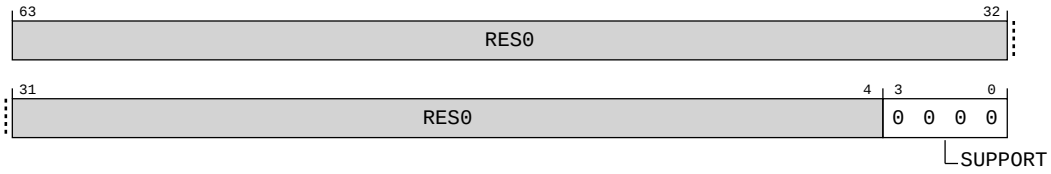


Table A-559: TRCIMSPEC0 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRCIMSPECn == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t, 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.15.5 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
```

63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3	0

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-217: AArch64_trcseqevr1 bit assignments

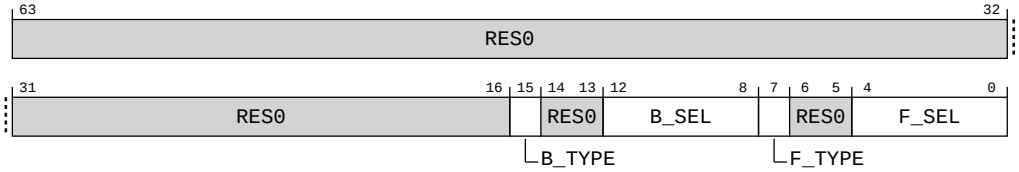


Table A-562: TRCSEQEVR1 bit descriptions

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**
[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.	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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.6 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

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-218: AArch64_trccntrldvr1 bit assignments

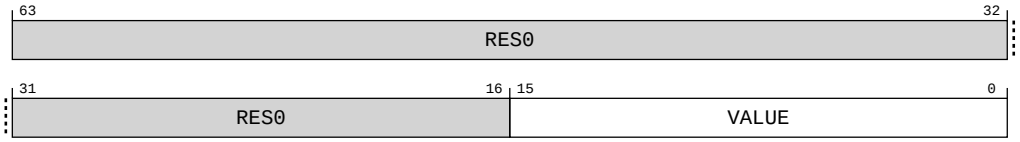


Table A-565: TRCCNTRLDVR1 bit descriptions

Bits	Name	Description	Reset
[63:16]	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.

MRS <Xt>, TRCCNTRLDVR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b101

MSR TRCCNTRLDVR1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	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>, TRCCNTRLDVR1

```
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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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] = TRCCNTRLDVR[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] = TRCCNTRLDVR[1];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCCNTRLDVR[1];
```

MSR TRCCNTRLDVR1, <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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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
        TRCCNTRLDVR[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
        TRCCNTRLDVR[1] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCNTRLDVR[1] = X[t, 64];
```

A.15.7 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-219: AArch64_trcseqevr2 bit assignments

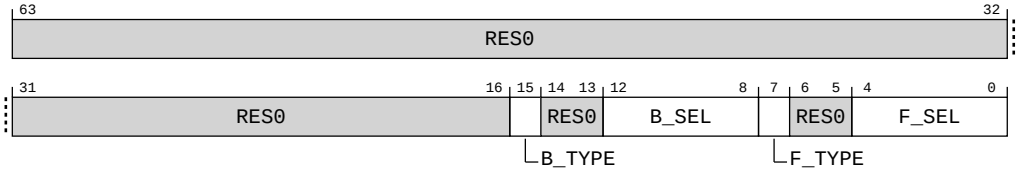


Table A-568: TRCSEQEVR2 bit descriptions

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.	`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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.8 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-220: AArch64_trcidr10 bit assignments

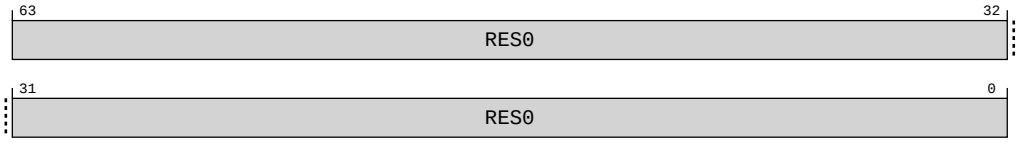


Table A-571: TRCIDR10 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.9 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-221: AArch64_trcidr11 bit assignments

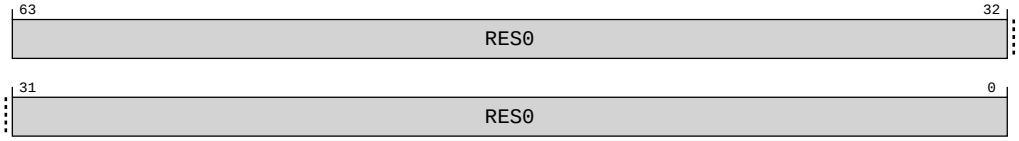


Table A-573: TRCIDR11 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.10 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-222: AArch64_trccntctlr0 bit assignments

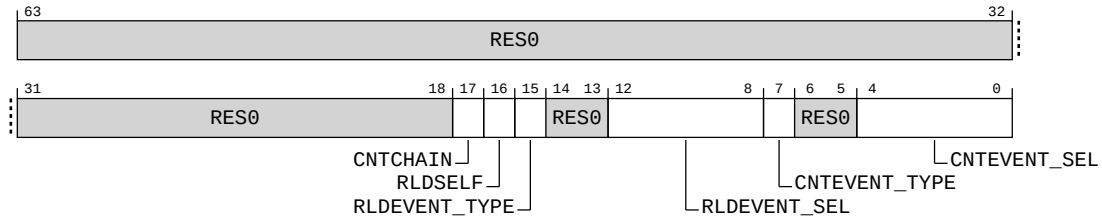


Table A-575: TRCCNTCTLR0 bit descriptions

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.	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>.	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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.11 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-223: AArch64_trcidr12 bit assignments

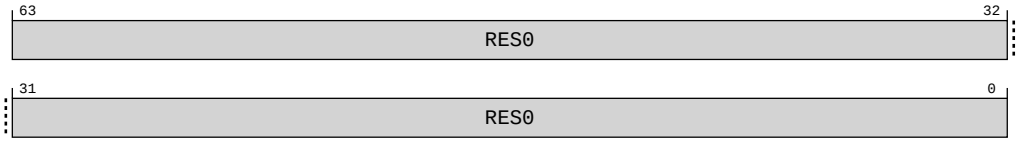


Table A-578: TRCIDR12 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.12 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-224: AArch64_trccntctlr1 bit assignments

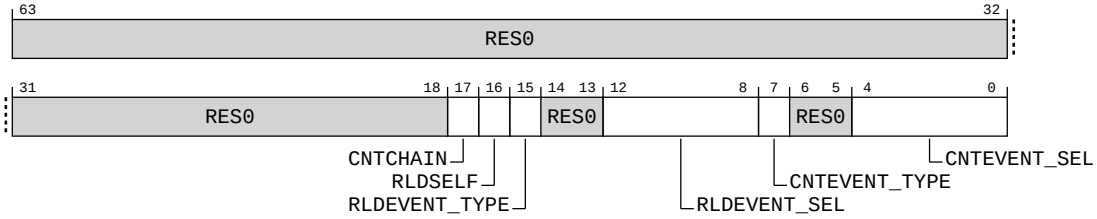


Table A-580: TRCCNTCTLR1 bit descriptions

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.	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>.	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**
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.13 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-225: AArch64_trcidr13 bit assignments

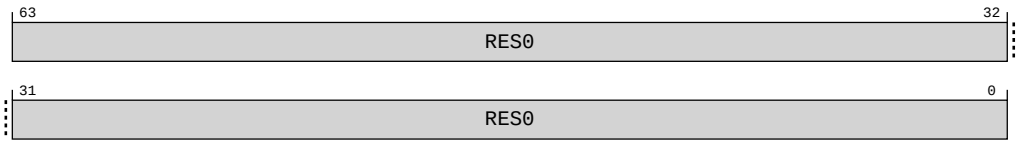


Table A-583: TRCIDR13 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.14 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-226: AArch64_trcextinselr0 bit assignments

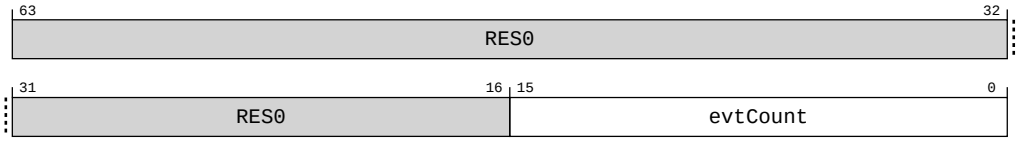


Table A-585: TRCEXTINSELR0 bit descriptions

Bits	Name	Description	Reset
[63:16]	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**.	`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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.15 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-227: AArch64_trccntvr0 bit assignments

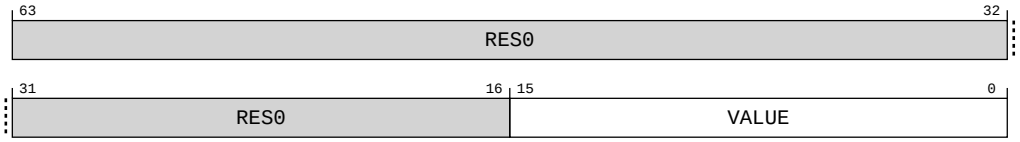


Table A-588: TRCCNTVR0 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCCNTVRn == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRCCNTVRn == '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.15.16 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 1xxx xxx0 00xx 111x 1010 000x
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-228: AArch64_trcidr0 bit assignments

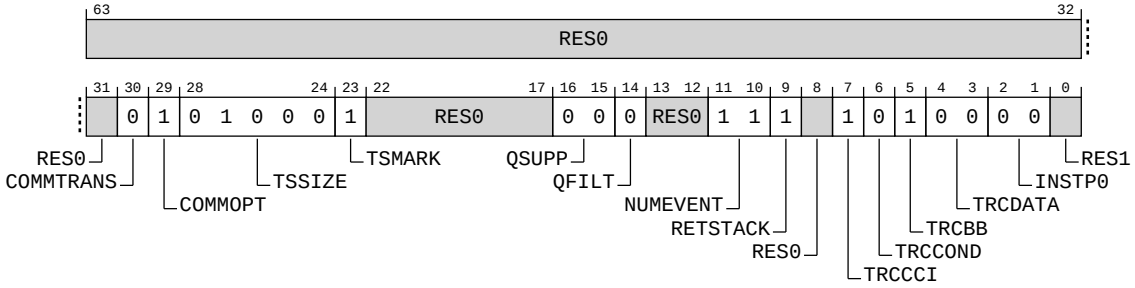


Table A-591: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	COMMTRANS	Transaction Start element behavior. **0b0** Transaction Start elements are P0 elements.	0b0
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **0b1** Commit mode 1. Access to this field is: **RAO/WI**	0b1
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **0b01000** Global timestamping implemented with a 64-bit timestamp value.	0b01000
[23]	TSMARK	Indicates whether Timestamp Marker elements are generated. **0b1** Timestamp Marker elements are generated.	0b1
[22:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	0b00
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	0b0
[13:12]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
```

```
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.17 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-229: AArch64_trcextinselr1 bit assignments

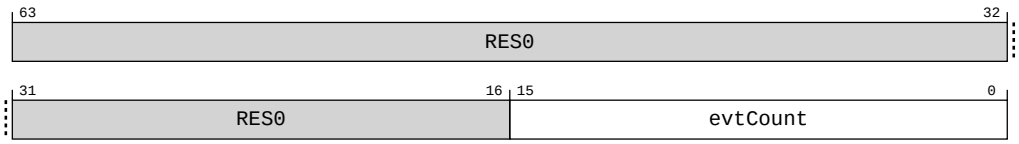


Table A-593: TRCEXTINSELR1 bit descriptions

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**.	`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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.18 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-230: AArch64_trccntvr1 bit assignments

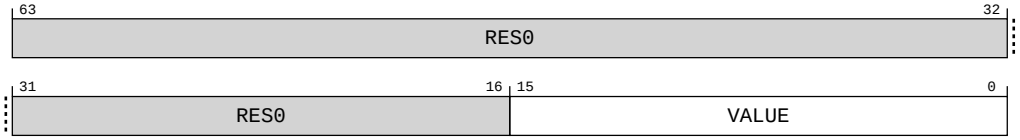


Table A-596: TRCCNTVR1 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCCNTVRn == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRCCNTVRn == '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.15.19 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 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-231: AArch64_trcidr1 bit assignments

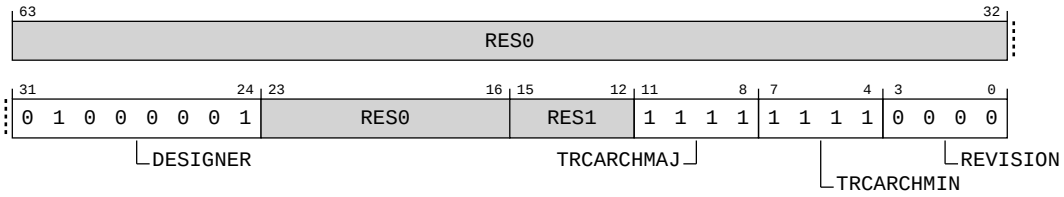


Table A-599: TRCIDR1 bit descriptions

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

Bits	Name	Description	Reset
[3:0]	REVISION	Implementation revision that identifies the revision of the trace and OS Lock registers. **0b0000** Revision 0	0b0000

Access

MRS <Xt>, TRCIDR1

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b111

Accessibility

MRS <Xt>, TRCIDR1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.20 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-232: AArch64_trcextinselr2 bit assignments

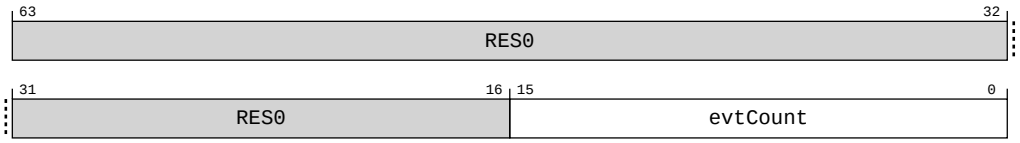


Table A-601: TRCEXTINSELR2 bit descriptions

Bits	Name	Description	Reset
[63:16]	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**.	`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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.21 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-233: AArch64_trcidr2 bit assignments

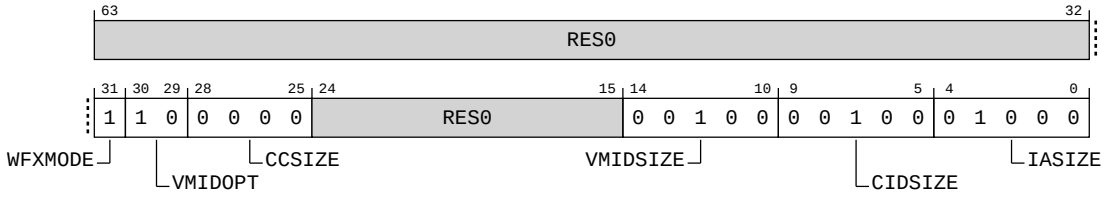


Table A-604: TRCIDR2 bit descriptions

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

Bits	Name	Description	Reset
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **0b10** Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is **RES1**.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.22 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-234: AArch64_trcextinselr3 bit assignments

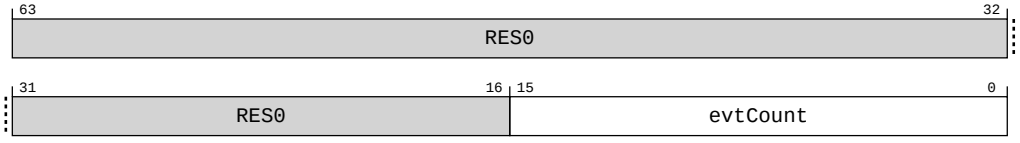


Table A-606: TRCEXTINSELR3 bit descriptions

Bits	Name	Description	Reset
[63:16]	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**.	`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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.23 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-235: AArch64_trcidr3 bit assignments

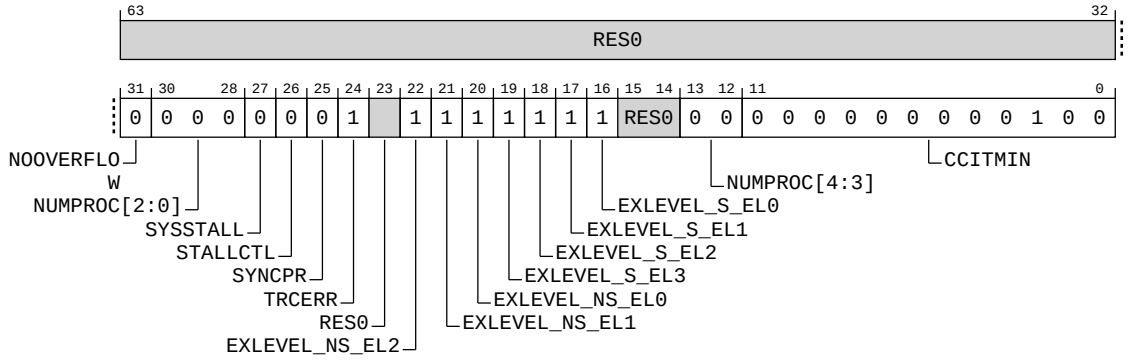


Table A-609: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	0b0
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **0b0** Stalling of the PE is not permitted.	0b0
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **0b0** Stalling of the PE is not implemented.	0b0
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0b0** 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
[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

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.24 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-236: AArch64_trcidr4 bit assignments

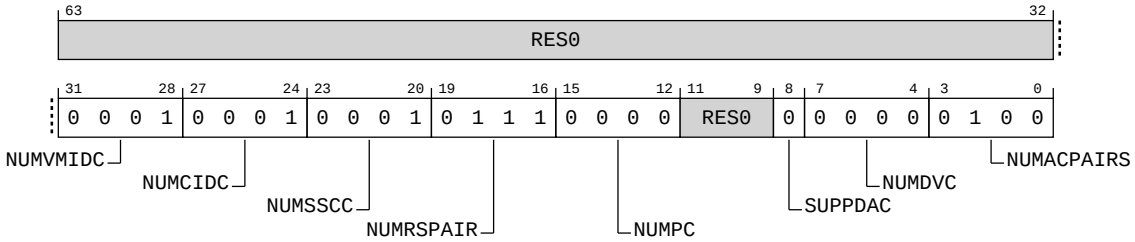


Table A-611: TRCIDR4 bit descriptions

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

Bits	Name	Description	Reset
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0b0001** The implementation has one Context Identifier Comparator.	0b0001
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0b0001** The implementation has one Single-shot Comparator Control.	0b0001
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0b0111** The implementation has eight resource selector pairs.	0b0111
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0b0000** No PE Comparator Inputs are available.	0b0000
[11:9]	**RES0**	Reserved	**RES0**
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0** Data address comparisons not implemented.	0b0
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures. **0b0000** No data value comparators implemented.	0b0000
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0b0100** The implementation has four Address Comparator pairs.	0b0100

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
```

```
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.25 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-237: AArch64_trcidr5 bit assignments

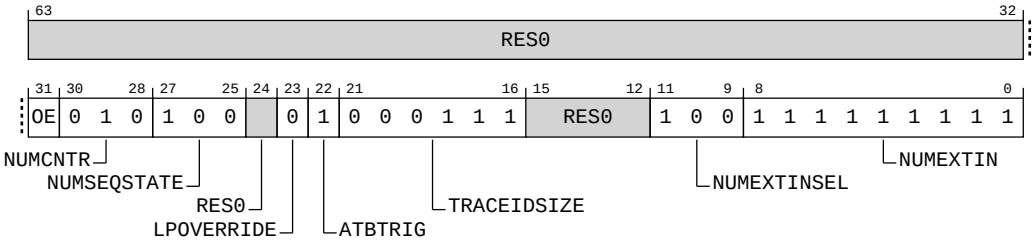


Table A-613: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	OE	Indicates support for the ETE Trace Output Enable. **0b0** ETE Trace Output Enable is not implemented. **0b1** ETE Trace Output Enable is implemented.	The reset values can be the following: `0b0`, `0b1`, respective to the value.
[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`

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        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.15.26 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-238: AArch64_trcssccr0 bit assignments

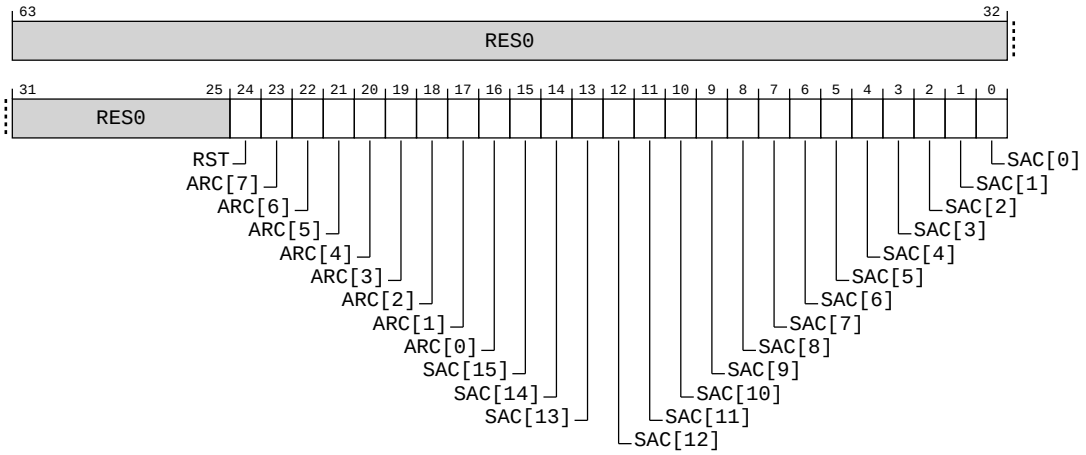


Table A-615: TRCSSCCR0 bit descriptions

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

Bits	Name	Description	Reset
[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.	8{x}
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.27 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-239: AArch64_trcrsctlr2 bit assignments

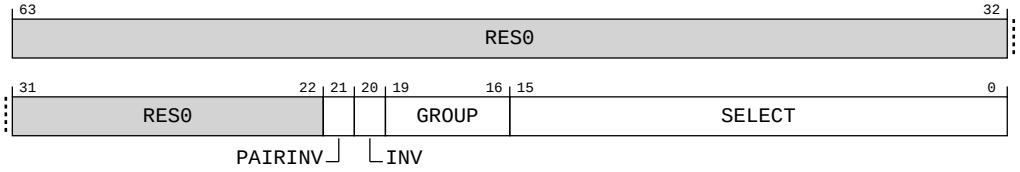


Table A-618: TRCRSCTLR2 bit descriptions

Bits	Name	Description	Reset
[63:22]	**RES0**	Reserved	**RES0**
[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.	x

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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.28 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-240: AArch64_trcrsctlr3 bit assignments

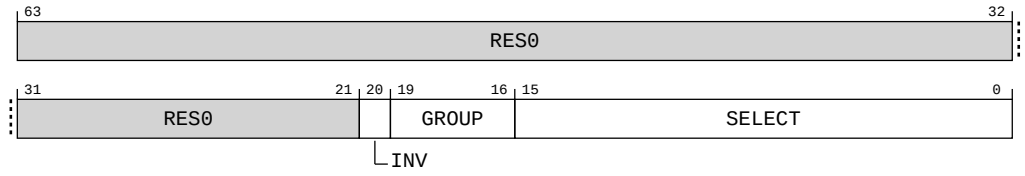


Table A-621: TRCRSCTLR3 bit descriptions

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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.29 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-241: AArch64_trcrsctlr4 bit assignments

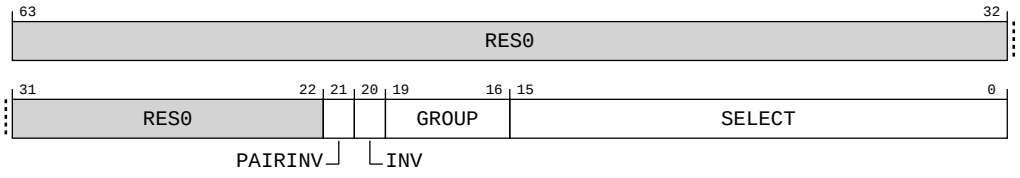


Table A-624: TRCRSCTLR4 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[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.	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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.30 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-242: AArch64_trcrsctlr5 bit assignments

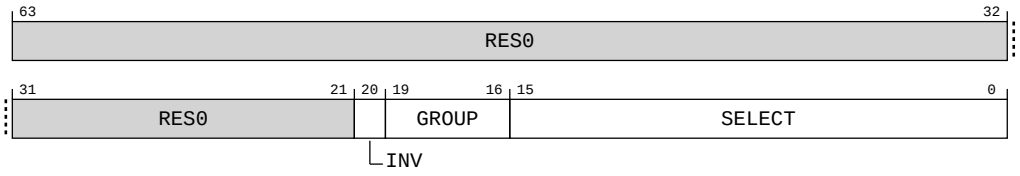


Table A-627: TRCRSCTLR5 bit descriptions

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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.31 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-243: AArch64_trcrsctlr6 bit assignments

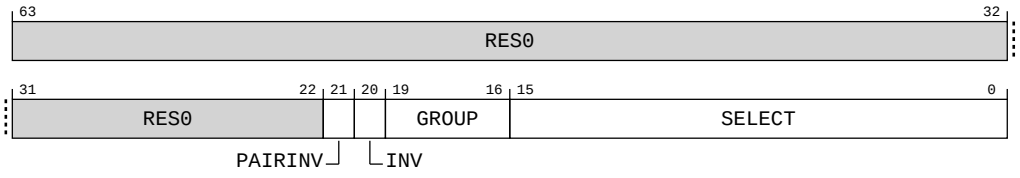


Table A-630: TRCRSCTLR6 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[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.	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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.32 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-244: AArch64_trcrsctlr7 bit assignments

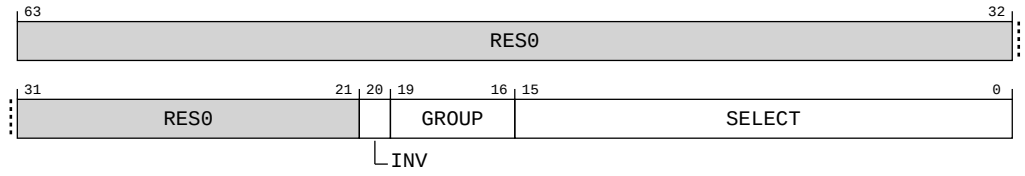


Table A-633: TRCRSCTLR7 bit descriptions

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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.33 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-245: AArch64_trcrsctlr8 bit assignments

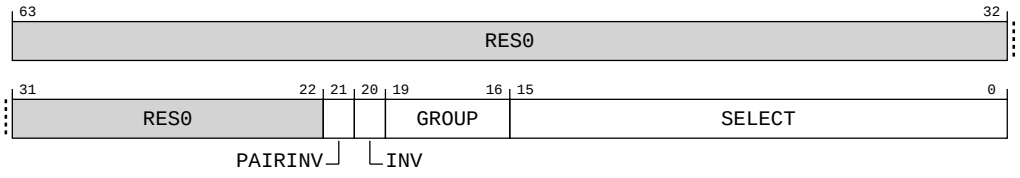


Table A-636: TRCRSCTLR8 bit descriptions

Bits	Name	Description	Reset
[63:22]	**RES0**	Reserved	**RES0**
[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.	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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.34 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-246: AArch64_trcsscsr0 bit assignments

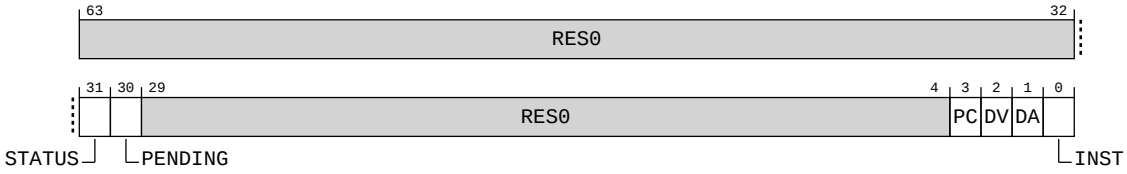


Table A-639: TRCSSCSR0 bit descriptions

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.	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.	x
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCSSCSRn == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRCSSCSRn == '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.15.35 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-247: AArch64_trcrsctlr9 bit assignments

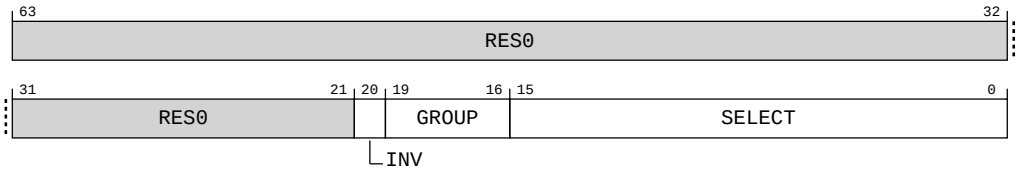


Table A-642: TRCRSCTLR9 bit descriptions

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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.36 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-248: AArch64_trcrsctlr10 bit assignments

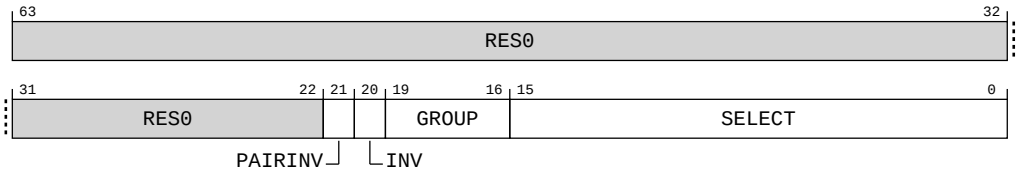


Table A-645: TRCRSCTLR10 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[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.	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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.37 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-249: AArch64_trcrsctlr11 bit assignments

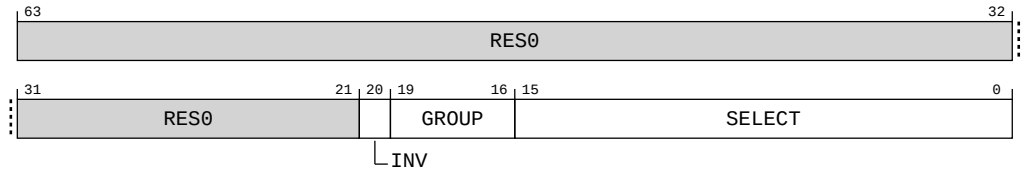


Table A-648: TRCRSCTLR11 bit descriptions

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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.38 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-250: AArch64_trcrsctlr12 bit assignments

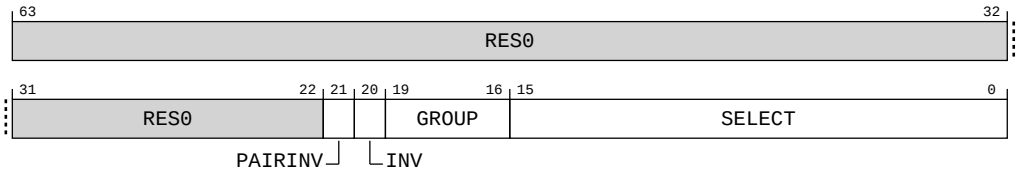


Table A-651: TRCRSCTLR12 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[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.	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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.39 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-251: AArch64_trcrsctlr13 bit assignments

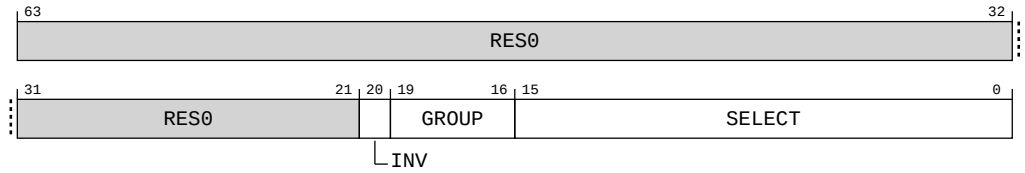


Table A-654: TRCRSCTLR13 bit descriptions

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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.40 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-252: AArch64_trcrsctlr14 bit assignments

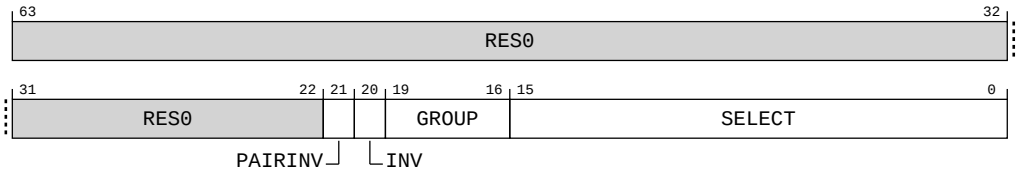


Table A-657: TRCRSCTLR14 bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[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.	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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.41 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-253: AArch64_trcrsctlr15 bit assignments

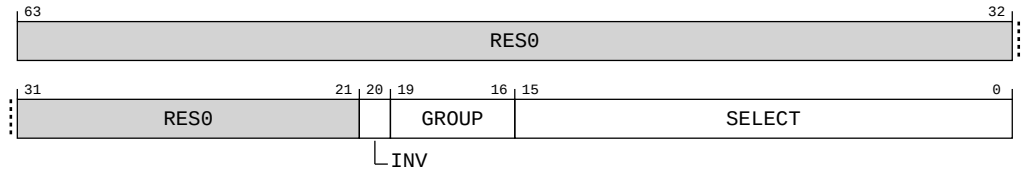


Table A-660: TRCRSCTLR15 bit descriptions

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
[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.	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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.42 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-254: AArch64_trcacvr0 bit assignments

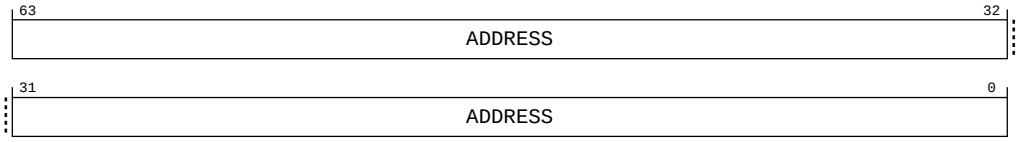


Table A-663: TRCACVR0 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.43 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-255: AArch64_trcacatr0 bit assignments

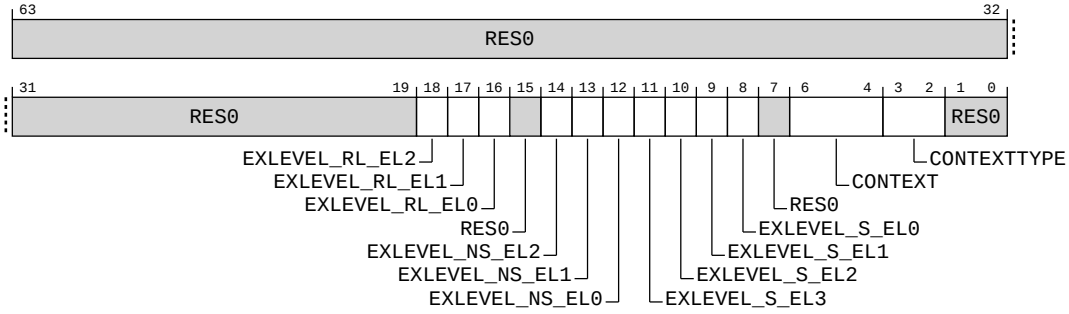


Table A-666: TRCACATR0 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.44 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-256: AArch64_trcacvr1 bit assignments

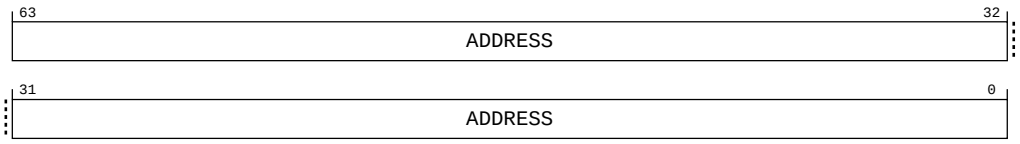


Table A-669: TRCACVR1 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.45 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-257: AArch64_trcacatr1 bit assignments

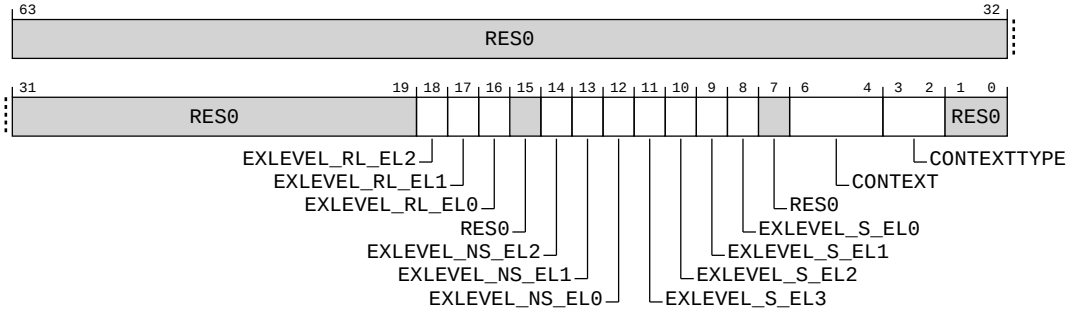


Table A-672: TRCACATR1 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.46 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-258: AArch64_trcacvr2 bit assignments

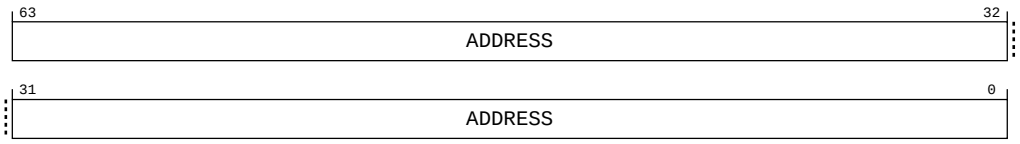


Table A-675: TRCACVR2 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.47 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-259: AArch64_trcacatr2 bit assignments

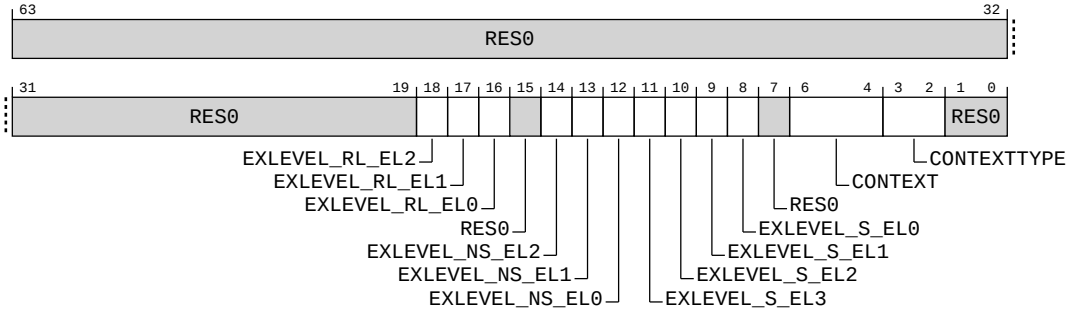


Table A-678: TRCACATR2 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.48 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-260: AArch64_trcacvr3 bit assignments

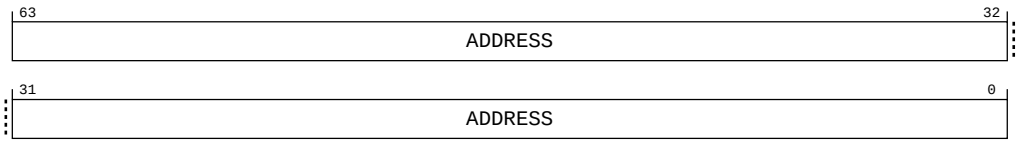


Table A-681: TRCACVR3 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.49 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-261: AArch64_trcacatr3 bit assignments

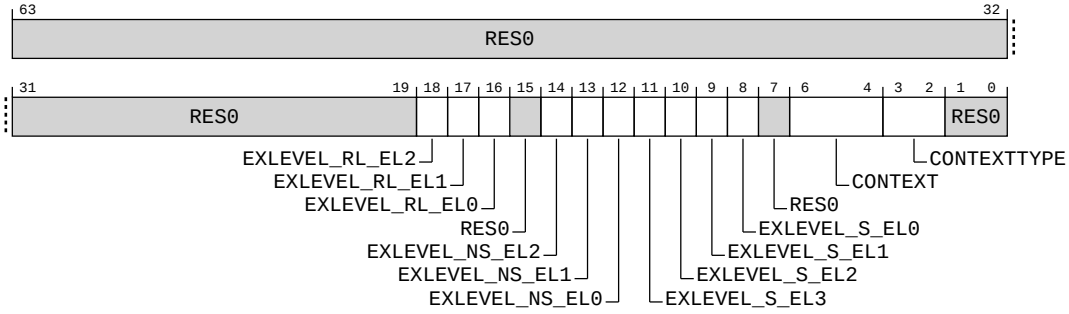


Table A-684: TRCACATR3 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.50 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-262: AArch64_trcacvr4 bit assignments

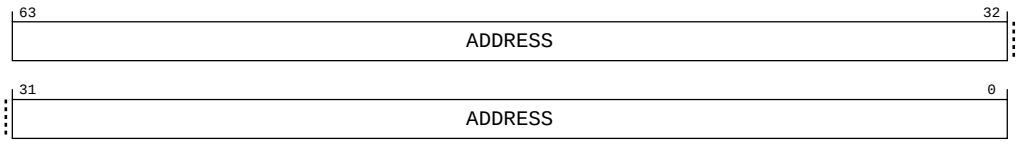


Table A-687: TRCACVR4 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.51 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-263: AArch64_trcacatr4 bit assignments

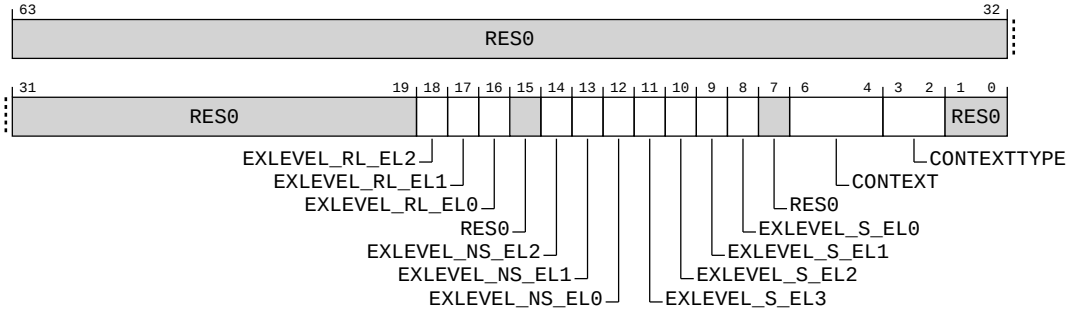


Table A-690: TRCACATR4 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.52 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-264: AArch64_trcacvr5 bit assignments

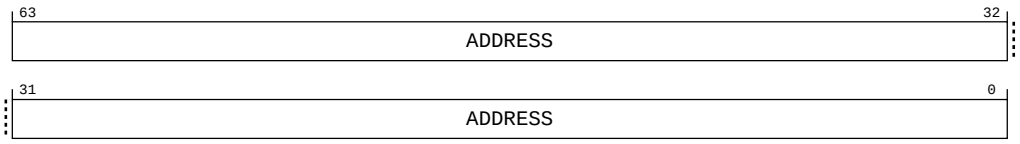


Table A-693: TRCACVR5 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.53 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-265: AArch64_trcacatr5 bit assignments

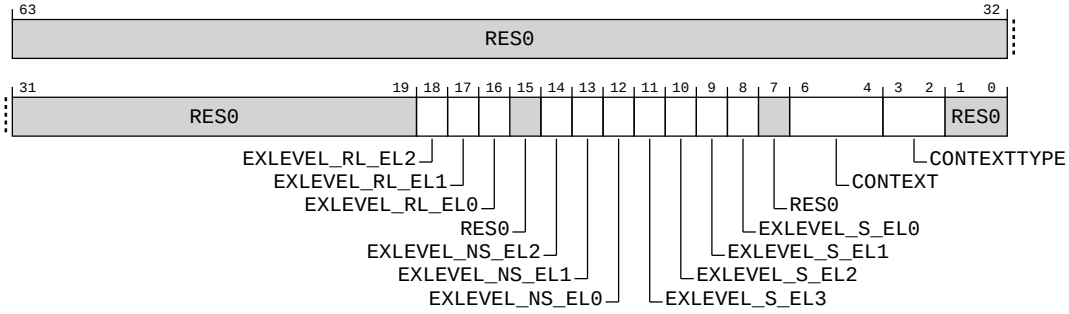


Table A-696: TRCACATR5 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.54 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-266: AArch64_trcacvr6 bit assignments

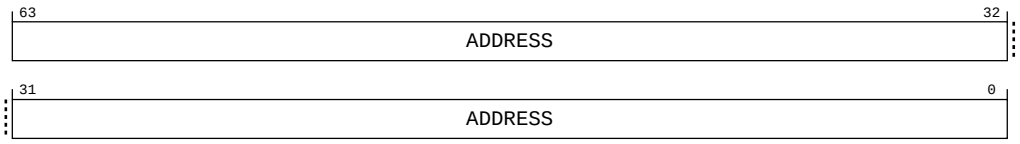


Table A-699: TRCACVR6 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.55 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-267: AArch64_trcacatr6 bit assignments

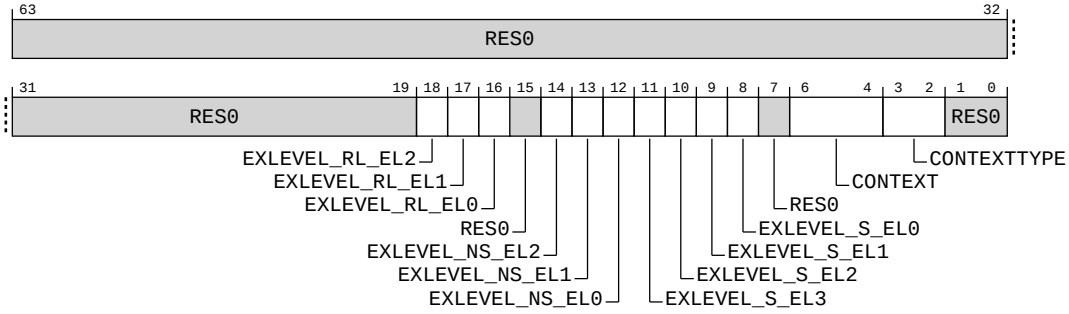


Table A-702: TRCACATR6 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.56 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-268: AArch64_trcacvr7 bit assignments

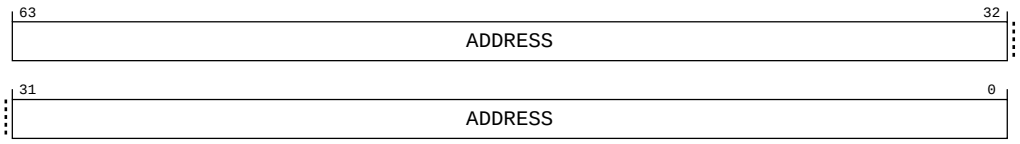


Table A-705: TRCACVR7 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.57 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-269: AArch64_trcacatr7 bit assignments

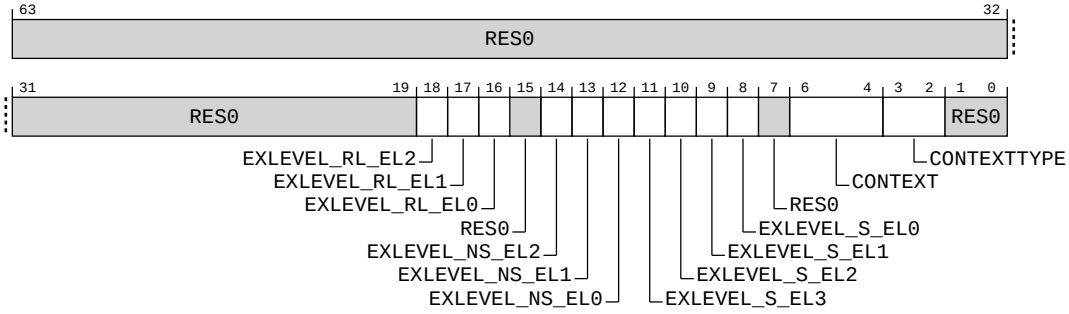


Table A-708: TRCACATR7 bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	EXLEVEL_RL_EL2	Realm EL2 address comparison control. Controls whether a comparison can occur at EL2 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator performs comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator does not perform comparisons in Realm EL2. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL2 is 0 the Address Comparator does not perform comparisons in Realm EL2. When TRCACATR<n>.EXLEVEL_NS_EL2 is 1 the Address Comparator performs comparisons in Realm EL2.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_RL_EL1	Realm EL1 address comparison control. Controls whether a comparison can occur at EL1 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator performs comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator does not perform comparisons in Realm EL1. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL1 is 0 the Address Comparator does not perform comparisons in Realm EL1. When TRCACATR<n>.EXLEVEL_NS_EL1 is 1 the Address Comparator performs comparisons in Realm EL1.	x
[16]	EXLEVEL_RL_EL0	Realm EL0 address comparison control. Controls whether a comparison can occur at EL0 in Realm state. **0b0** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator performs comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator does not perform comparisons in Realm EL0. **0b1** When TRCACATR<n>.EXLEVEL_NS_EL0 is 0 the Address Comparator does not perform comparisons in Realm EL0. When TRCACATR<n>.EXLEVEL_NS_EL0 is 1 the Address Comparator performs comparisons in Realm EL0.	x
[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

Bits	Name	Description	Reset
[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
[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.	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.	xx

Bits	Name	Description	Reset
[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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.58 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-270: AArch64_trccidcvr0 bit assignments

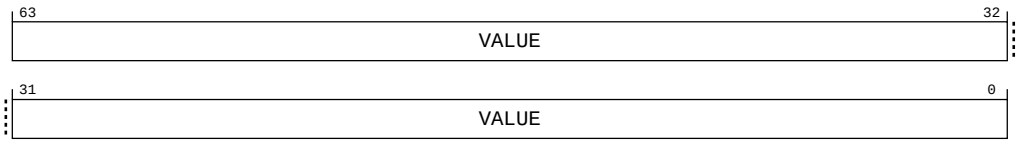


Table A-711: TRCCIDCVR0 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.15.59 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-271: AArch64_trcvmidcvr0 bit assignments

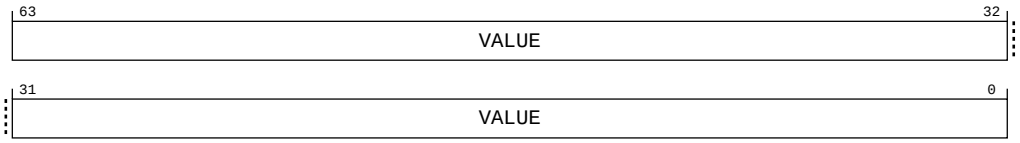


Table A-714: TRCVMIDCVR0 bit descriptions

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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRC == '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 EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.TRC == '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.16 AArch64 Memory Partitioning and Monitoring registers summary

The following summary table provides an overview of all Memory Partitioning and Monitoring registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-717: Memory Partitioning and Monitoring registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MPAM1_EL1	3	0	C10	C5	0	See individual bit resets.	64-bit	MPAM1 Register (EL1)
MPAM0_EL1	3	0	C10	C5	1	See individual bit resets.	64-bit	MPAM0 Register (EL1)
MPAMHCR_EL2	3	4	C10	C4	0	See individual bit resets.	64-bit	MPAM Hypervisor Control Register (EL2)
MPAMVPMV_EL2	3	4	C10	C4	1	See individual bit resets.	64-bit	MPAM Virtual Partition Mapping Valid Register
MPAM2_EL2	3	4	C10	C5	0	See individual bit resets.	64-bit	MPAM2 Register (EL2)
MPAMVPM0_EL2	3	4	C10	C6	0	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 0
MPAMVPM1_EL2	3	4	C10	C6	1	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 1
MPAMVPM2_EL2	3	4	C10	C6	2	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 2
MPAMVPM3_EL2	3	4	C10	C6	3	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 3
MPAMVPM4_EL2	3	4	C10	C6	4	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 4
MPAMVPM5_EL2	3	4	C10	C6	5	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 5
MPAMVPM6_EL2	3	4	C10	C6	6	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 6
MPAMVPM7_EL2	3	4	C10	C6	7	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 7

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MPAM3_EL3	3	6	C10	C5	0	See individual bit resets.	64-bit	MPAM3 Register (EL3)

A.16.1 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-272: AArch64_mpamvpmv_el2 bit assignments

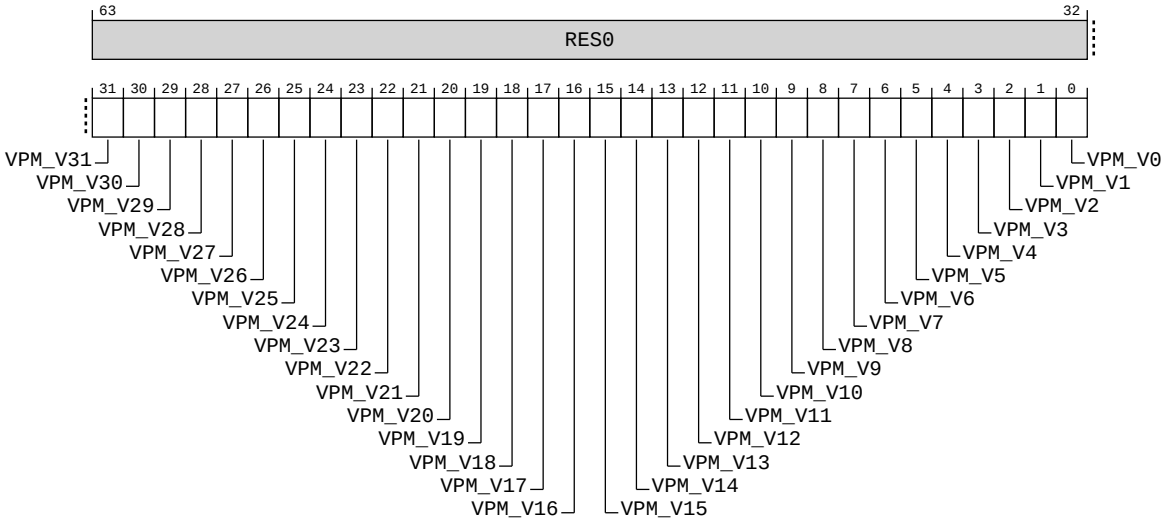


Table A-718: MPAMVPMV_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	VPM_V<m>, bit[m], where m = 31 to 0	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	32{x}

Access

MRS <Xt>, MPAMVPMV_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

MSR MPAMVPMV_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0100	0b001

Accessibility

MRS <Xt>, MPAMVPMV_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x938];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
```

```
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if 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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x938] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if 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.16.2 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-273: AArch64_mpamvpm0_el2 bit assignments

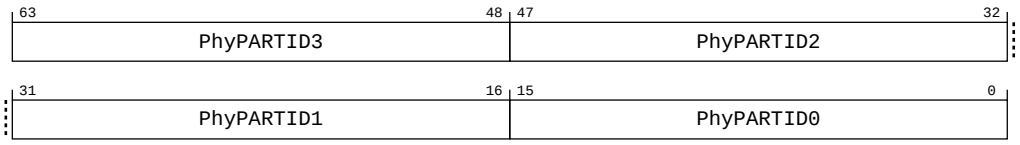


Table A-721: MPAMVPM0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID3	Virtual PARTID Mapping Entry for virtual PARTID 3. PhyPARTID3 gives the mapping of virtual PARTID 3 to a physical PARTID.	16{x}
[47:32]	PhyPARTID2	Virtual PARTID Mapping Entry for virtual PARTID 2. PhyPARTID2 gives the mapping of virtual PARTID 2 to a physical PARTID.	16{x}
[31:16]	PhyPARTID1	Virtual PARTID Mapping Entry for virtual PARTID 1. PhyPARTID1 gives the mapping of virtual PARTID 1 to a physical PARTID.	16{x}
[15:0]	PhyPARTID0	Virtual PARTID Mapping Entry for virtual PARTID 0. PhyPARTID0 gives the mapping of virtual PARTID 0 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM0_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

MSR MPAMVPM0_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b000

Accessibility

MRS <Xt>, MPAMVPM0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x940];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if 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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x940] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if 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.16.3 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-274: AArch64_mpamvpm1_el2 bit assignments

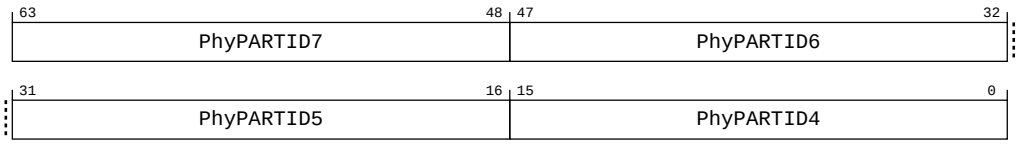


Table A-724: MPAMVPM1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID7	Virtual PARTID Mapping Entry for virtual PARTID 7. PhyPARTID7 gives the mapping of virtual PARTID 7 to a physical PARTID.	16{x}
[47:32]	PhyPARTID6	Virtual PARTID Mapping Entry for virtual PARTID 6. PhyPARTID6 gives the mapping of virtual PARTID 6 to a physical PARTID.	16{x}
[31:16]	PhyPARTID5	Virtual PARTID Mapping Entry for virtual PARTID 5. PhyPARTID5 gives the mapping of virtual PARTID 5 to a physical PARTID.	16{x}
[15:0]	PhyPARTID4	Virtual PARTID Mapping Entry for virtual PARTID 4. PhyPARTID4 gives the mapping of virtual PARTID 4 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM1_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

MSR MPAMVPM1_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b001

Accessibility

MRS <Xt>, MPAMVPM1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x948];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if 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
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x948] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if 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.16.4 MPAMVPM2_EL2, MPAM Virtual PARTID Mapping Register 2

MPAMVPM2_EL2 provides mappings from virtual PARTIDs 8 - 11 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented AArch64-MPAMVPM0_EL2 to AArch64-MPAMVPM7_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-275: AArch64_mpamvpm2_el2 bit assignments

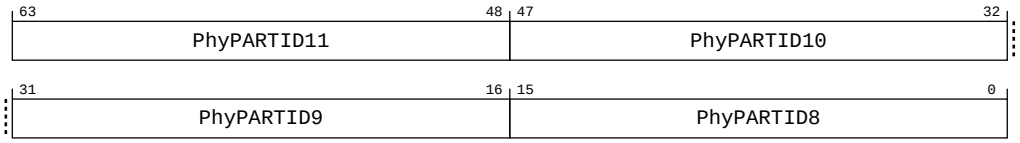


Table A-727: MPAMVPM2_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID11	Virtual PARTID Mapping Entry for virtual PARTID 11. PhyPARTID11 gives the mapping of virtual PARTID 11 to a physical PARTID.	16{x}
[47:32]	PhyPARTID10	Virtual PARTID Mapping Entry for virtual PARTID 10. PhyPARTID10 gives the mapping of virtual PARTID 10 to a physical PARTID.	16{x}
[31:16]	PhyPARTID9	Virtual PARTID Mapping Entry for virtual PARTID 9. PhyPARTID9 gives the mapping of virtual PARTID 9 to a physical PARTID.	16{x}
[15:0]	PhyPARTID8	Virtual PARTID Mapping Entry for virtual PARTID 8. PhyPARTID8 gives the mapping of virtual PARTID 8 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM2_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b010

MSR MPAMVPM2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b010

Accessibility

MRS <Xt>, MPAMVPM2_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x950];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM2_EL2;
```

MSR MPAMVPM2_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x950] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM2_EL2 = X[t, 64];
```

A.16.5 MPAMVPM3_EL2, MPAM Virtual PARTID Mapping Register 3

MPAMVPM3_EL2 provides mappings from virtual PARTIDs 12 - 15 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-276: AArch64_mpamvpm3_el2 bit assignments

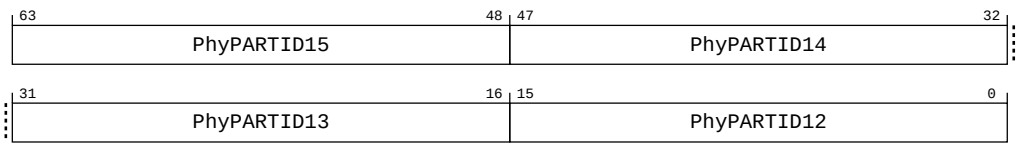


Table A-730: MPAMVPM3_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID15	Virtual PARTID Mapping Entry for virtual PARTID 15. PhyPARTID15 gives the mapping of virtual PARTID 15 to a physical PARTID.	16{x}
[47:32]	PhyPARTID14	Virtual PARTID Mapping Entry for virtual PARTID 14. PhyPARTID14 gives the mapping of virtual PARTID 14 to a physical PARTID.	16{x}
[31:16]	PhyPARTID13	Virtual PARTID Mapping Entry for virtual PARTID 13. PhyPARTID13 gives the mapping of virtual PARTID 13 to a physical PARTID.	16{x}
[15:0]	PhyPARTID12	Virtual PARTID Mapping Entry for virtual PARTID 12. PhyPARTID12 gives the mapping of virtual PARTID 12 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM3_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b011

MSR MPAMVPM3_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b011

Accessibility

MRS <Xt>, MPAMVPM3_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x958];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM3_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM3_EL2;
```

MSR MPAMVPM3_EL2, <Xt>

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

```
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x958] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM3_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM3_EL2 = X[t, 64];
```

A.16.6 MPAMVPM4_EL2, MPAM Virtual PARTID Mapping Register 4

MPAMVPM4_EL2 provides mappings from virtual PARTIDs 16 - 19 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-277: AArch64_mpamvpm4_el2 bit assignments

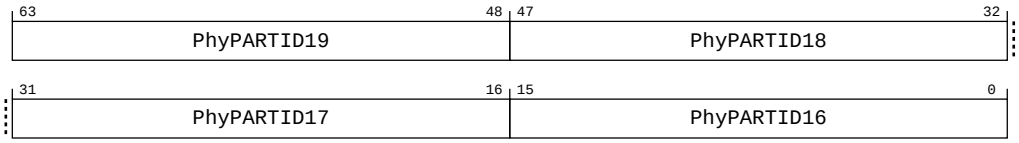


Table A-733: MPAMVPM4_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID19	Virtual PARTID Mapping Entry for virtual PARTID 19. PhyPARTID19 gives the mapping of virtual PARTID 19 to a physical PARTID.	16{x}
[47:32]	PhyPARTID18	Virtual PARTID Mapping Entry for virtual PARTID 18. PhyPARTID18 gives the mapping of virtual PARTID 18 to a physical PARTID.	16{x}
[31:16]	PhyPARTID17	Virtual PARTID Mapping Entry for virtual PARTID 17. PhyPARTID17 gives the mapping of virtual PARTID 17 to a physical PARTID.	16{x}
[15:0]	PhyPARTID16	Virtual PARTID Mapping Entry for virtual PARTID 16. PhyPARTID16 gives the mapping of virtual PARTID 16 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM4_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b100

MSR MPAMVPM4_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b100

Accessibility

MRS <Xt>, MPAMVPM4_EL2

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

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x960];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM4_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM4_EL2;
```

MSR MPAMVPM4_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x960] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM4_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM4_EL2 = X[t, 64];
```

A.16.7 MPAMVPM5_EL2, MPAM Virtual PARTID Mapping Register 5

MPAMVPM5_EL2 provides mappings from virtual PARTIDs 20 - 23 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-278: AArch64_mpamvpm5_el2 bit assignments

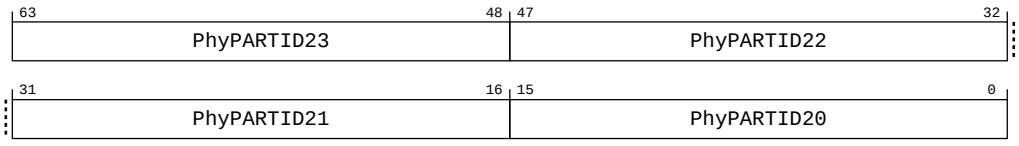


Table A-736: MPAMVPM5_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID23	Virtual PARTID Mapping Entry for virtual PARTID 23. PhyPARTID23 gives the mapping of virtual PARTID 23 to a physical PARTID.	16{x}
[47:32]	PhyPARTID22	Virtual PARTID Mapping Entry for virtual PARTID 22. PhyPARTID22 gives the mapping of virtual PARTID 22 to a physical PARTID.	16{x}
[31:16]	PhyPARTID21	Virtual PARTID Mapping Entry for virtual PARTID 21. PhyPARTID21 gives the mapping of virtual PARTID 21 to a physical PARTID.	16{x}

Bits	Name	Description	Reset
[15:0]	PhyPARTID20	Virtual PARTID Mapping Entry for virtual PARTID 20. PhyPARTID20 gives the mapping of virtual PARTID 20 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM5_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b101

MSR MPAMVPM5_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b101

Accessibility

MRS <Xt>, MPAMVPM5_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x968];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM5_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM5_EL2;
```

MSR MPAMVPM5_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x968] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
```

```
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM5_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM5_EL2 = X[t, 64];
```

A.16.8 MPAMVPM6_EL2, MPAM Virtual PARTID Mapping Register 6

MPAMVPM6_EL2 provides mappings from virtual PARTIDs 24 - 27 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for PARTIDs in AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-279: AArch64_mpamvpm6_el2 bit assignments

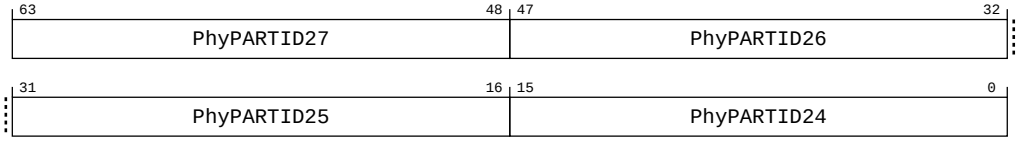


Table A-739: MPAMVPM6_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID27	Virtual PARTID Mapping Entry for virtual PARTID 27. PhyPARTID27 gives the mapping of virtual PARTID 27 to a physical PARTID.	16{x}
[47:32]	PhyPARTID26	Virtual PARTID Mapping Entry for virtual PARTID 26. PhyPARTID26 gives the mapping of virtual PARTID 26 to a physical PARTID.	16{x}
[31:16]	PhyPARTID25	Virtual PARTID Mapping Entry for virtual PARTID 25. PhyPARTID25 gives the mapping of virtual PARTID 25 to a physical PARTID.	16{x}
[15:0]	PhyPARTID24	Virtual PARTID Mapping Entry for virtual PARTID 24. PhyPARTID24 gives the mapping of virtual PARTID 24 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM6_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b110

MSR MPAMVPM6_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b110

Accessibility

MRS <Xt>, MPAMVPM6_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x970];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
```

```
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM6_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM6_EL2;
```

MSR MPAMVPM6_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x970] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM6_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MPAMVPM6_EL2 = X[t, 64];
```

A.16.9 MPAMVPM7_EL2, MPAM Virtual PARTID Mapping Register 7

MPAMVPM7_EL2 provides mappings from virtual PARTIDs 28 - 31 to physical PARTIDs.

AArch64-MPAMIDR_EL1.VPMR_MAX field gives the index of the highest implemented MPAMVPM<n>_EL2 registers. VPMR_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR_EL1.VPMR_MAX == 0, there is only a single MPAMVPM<n>_EL2 register, AArch64-MPAMVPM0_EL2.

Virtual PARTID mapping is enabled by AArch64-MPAMHCR_EL2.EL1_VPMEN for PARTIDs in AArch64-MPAM1_EL1 and by AArch64-MPAMHCR_EL2.EL0_VPMEN for AArch64-MPAM0_EL1.

A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-280: AArch64_mpamvpm7_el2 bit assignments

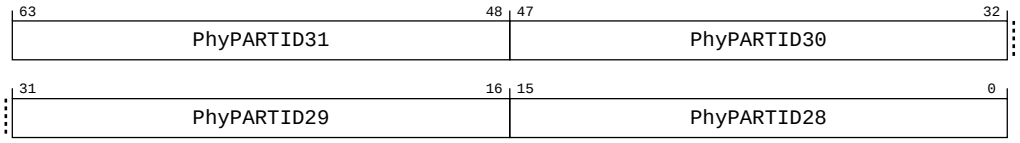


Table A-742: MPAMVPM7_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	PhyPARTID31	Virtual PARTID Mapping Entry for virtual PARTID 31. PhyPARTID31 gives the mapping of virtual PARTID 31 to a physical PARTID.	16{x}
[47:32]	PhyPARTID30	Virtual PARTID Mapping Entry for virtual PARTID 30. PhyPARTID30 gives the mapping of virtual PARTID 30 to a physical PARTID.	16{x}
[31:16]	PhyPARTID29	Virtual PARTID Mapping Entry for virtual PARTID 29. PhyPARTID29 gives the mapping of virtual PARTID 29 to a physical PARTID.	16{x}
[15:0]	PhyPARTID28	Virtual PARTID Mapping Entry for virtual PARTID 28. PhyPARTID28 gives the mapping of virtual PARTID 28 to a physical PARTID.	16{x}

Access

MRS <Xt>, MPAMVPM7_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b111

MSR MPAMVPM7_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0110	0b111

Accessibility

MRS <Xt>, MPAMVPM7_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x978];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MPAMVPM7_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MPAMVPM7_EL2;
```

MSR MPAMVPM7_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x978] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM7_EL2 = X[t, 64];
```

```
elsif PSTATE.EL == EL3 then
    MPAMVPM7_EL2 = X[t, 64];
```

A.17 AArch64 Statistical Profiling Extension registers summary

The following summary table provides an overview of all Statistical Profiling Extension registers in the core.

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-745: Statistical Profiling Extension registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMSCR_EL1	3	0	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL1)
PMSNEVFR_EL1	3	0	C9	C9	1	See individual bit resets.	64-bit	Sampling Inverted Event Filter Register
PMSICR_EL1	3	0	C9	C9	2	See individual bit resets.	64-bit	Sampling Interval Counter Register
PMSIRR_EL1	3	0	C9	C9	3	See individual bit resets.	64-bit	Sampling Interval Reload Register
PMSFCR_EL1	3	0	C9	C9	4	See individual bit resets.	64-bit	Sampling Filter Control Register
PMSEVFR_EL1	3	0	C9	C9	5	See individual bit resets.	64-bit	Sampling Event Filter Register
PMSLATFR_EL1	3	0	C9	C9	6	See individual bit resets.	64-bit	Sampling Latency Filter Register
PMSIDR_EL1	3	0	C9	C9	7	See individual bit resets.	64-bit	Sampling Profiling ID Register
PMBLIMITR_EL1	3	0	C9	C10	0	See individual bit resets.	64-bit	Profiling Buffer Limit Address Register
PMBPTR_EL1	3	0	C9	C10	1	See individual bit resets.	64-bit	Profiling Buffer Write Pointer Register
PMBSR_EL1	3	0	C9	C10	3	See individual bit resets.	64-bit	Profiling Buffer Status/syndrome Register
PMBIDR_EL1	3	0	C9	C10	7	See individual bit resets.	64-bit	Profiling Buffer ID Register
PMSCR_EL2	3	4	C9	C9	0	See individual bit resets.	64-bit	Statistical Profiling Control Register (EL2)

A.17.1 PMSNEVFR_EL1, Sampling Inverted Event Filter Register

Controls sample filtering by events. The overall inverted filter is the logical OR of these filters. For example, if PMSNEVFR_EL1.E[3] and PMSNEVFR_EL1.E[5] are both set to 1, samples that have either event 3 (Level 1 unified or data cache refill) or event 5 (TLB walk) set to 1 are not recorded.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Statistical Profiling Extension registers

Access type

See bit descriptions

Reset value

```
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0xx0 0000 xxxx xxxx xxx0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-281: AArch64_pmsnevfr_el1 bit assignments

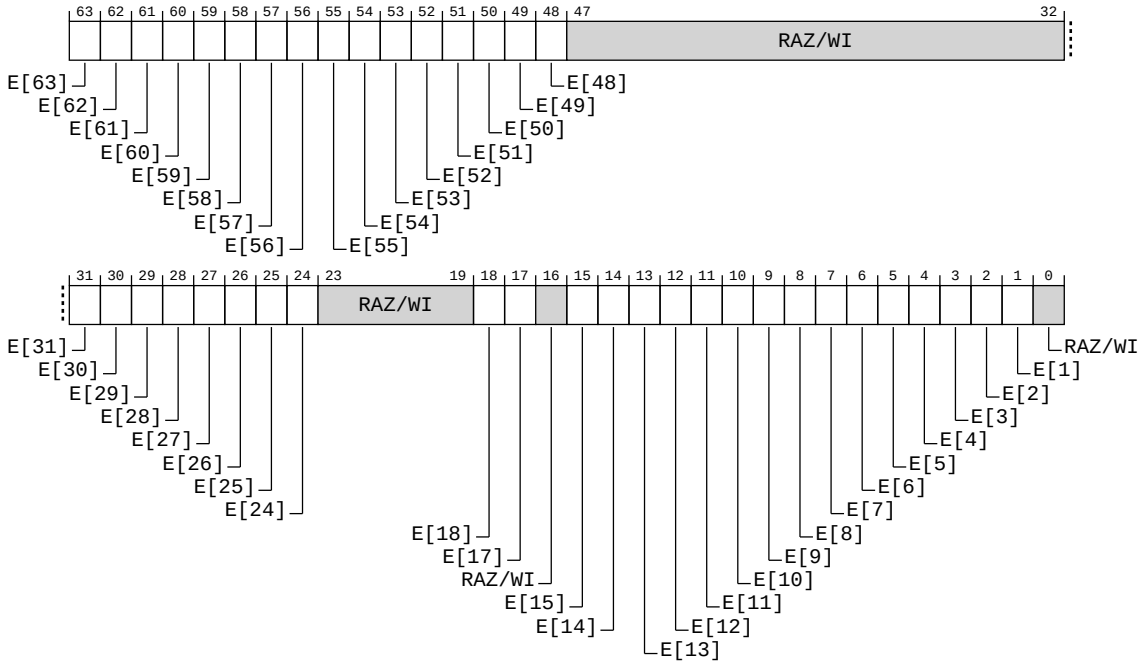


Table A-746: PMSNEVFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	E[63]	E[63] is the event filter for **IMPLEMENTATION DEFINED** event 63. **0b0** Event 63 is ignored. **0b1** Do not record samples that have event 63 == 1. Access to this field is: **RAZ/WI**	0b0
[62]	E[62]	E[62] is the event filter for **IMPLEMENTATION DEFINED** event 62. **0b0** Event 62 is ignored. **0b1** Do not record samples that have event 62 == 1. Access to this field is: **RAZ/WI**	0b0
[61]	E[61]	E[61] is the event filter for **IMPLEMENTATION DEFINED** event 61. **0b0** Event 61 is ignored. **0b1** Do not record samples that have event 61 == 1. Access to this field is: **RAZ/WI**	0b0
[60]	E[60]	E[60] is the event filter for **IMPLEMENTATION DEFINED** event 60. **0b0** Event 60 is ignored. **0b1** Do not record samples that have event 60 == 1. Access to this field is: **RAZ/WI**	0b0
[59]	E[59]	E[59] is the event filter for **IMPLEMENTATION DEFINED** event 59. **0b0** Event 59 is ignored. **0b1** Do not record samples that have event 59 == 1. Access to this field is: **RAZ/WI**	0b0
[58]	E[58]	E[58] is the event filter for **IMPLEMENTATION DEFINED** event 58. **0b0** Event 58 is ignored. **0b1** Do not record samples that have event 58 == 1. Access to this field is: **RAZ/WI**	0b0

Bits	Name	Description	Reset
[57]	E[57]	E[57] is the event filter for **IMPLEMENTATION DEFINED** event 57. **0b0** Event 57 is ignored. **0b1** Do not record samples that have event 57 == 1. Access to this field is: **RAZ/WI**	0b0
[56]	E[56]	E[56] is the event filter for **IMPLEMENTATION DEFINED** event 56. **0b0** Event 56 is ignored. **0b1** Do not record samples that have event 56 == 1. Access to this field is: **RAZ/WI**	0b0
[55]	E[55]	E[55] is the event filter for **IMPLEMENTATION DEFINED** event 55. **0b0** Event 55 is ignored. **0b1** Do not record samples that have event 55 == 1. Access to this field is: **RAZ/WI**	0b0
[54]	E[54]	E[54] is the event filter for **IMPLEMENTATION DEFINED** event 54. **0b0** Event 54 is ignored. **0b1** Do not record samples that have event 54 == 1. Access to this field is: **RAZ/WI**	0b0
[53]	E[53]	E[53] is the event filter for **IMPLEMENTATION DEFINED** event 53. **0b0** Event 53 is ignored. **0b1** Do not record samples that have event 53 == 1. Access to this field is: **RAZ/WI**	0b0
[52]	E[52]	E[52] is the event filter for **IMPLEMENTATION DEFINED** event 52. **0b0** Event 52 is ignored. **0b1** Do not record samples that have event 52 == 1. Access to this field is: **RAZ/WI**	0b0

Bits	Name	Description	Reset
[51]	E[51]	E[51] is the event filter for **IMPLEMENTATION DEFINED** event 51. **0b0** Event 51 is ignored. **0b1** Do not record samples that have event 51 == 1. Access to this field is: **RAZ/WI**	0b0
[50]	E[50]	E[50] is the event filter for **IMPLEMENTATION DEFINED** event 50. **0b0** Event 50 is ignored. **0b1** Do not record samples that have event 50 == 1. Access to this field is: **RAZ/WI**	0b0
[49]	E[49]	E[49] is the event filter for **IMPLEMENTATION DEFINED** event 49. **0b0** Event 49 is ignored. **0b1** Do not record samples that have event 49 == 1. Access to this field is: **RAZ/WI**	0b0
[48]	E[48]	E[48] is the event filter for **IMPLEMENTATION DEFINED** event 48. **0b0** Event 48 is ignored. **0b1** Do not record samples that have event 48 == 1. Access to this field is: **RAZ/WI**	0b0
[47:32]	**RAZ/WI**	Reserved	**RAZ/WI**
[31]	E[31]	E[31] is the event filter for **IMPLEMENTATION DEFINED** event 31. **0b0** Event 31 is ignored. **0b1** Do not record samples that have event 31 == 1. Access to this field is: **RAZ/WI**	0b0
[30]	E[30]	E[30] is the event filter for **IMPLEMENTATION DEFINED** event 30. **0b0** Event 30 is ignored. **0b1** Do not record samples that have event 30 == 1. Access to this field is: **RAZ/WI**	0b0

Bits	Name	Description	Reset
[29]	E[29]	E[29] is the event filter for **IMPLEMENTATION DEFINED** event 29. **0b0** Event 29 is ignored. **0b1** Do not record samples that have event 29 == 1. Access to this field is: **RAZ/WI**	0b0
[28]	E[28]	E[28] is the event filter for **IMPLEMENTATION DEFINED** event 28. **0b0** Event 28 is ignored. **0b1** Do not record samples that have event 28 == 1. Access to this field is: **RAZ/WI**	0b0
[27]	E[27]	E[27] is the event filter for **IMPLEMENTATION DEFINED** event 27. **0b0** Event 27 is ignored. **0b1** Do not record samples that have event 27 == 1. Access to this field is: **RAZ/WI**	0b0
[26]	E[26]	E[26] is the event filter for **IMPLEMENTATION DEFINED** event 26. **0b0** Event 26 is ignored. **0b1** Do not record samples that have event 26 == 1. Access to this field is: **RAZ/WI**	0b0
[25]	E[25]	E[25] is the event filter for **IMPLEMENTATION DEFINED** event 25. **0b0** Event 25 is ignored. **0b1** Do not record samples that have event 25 == 1. Access to this field is: **RAZ/WI**	0b0
[24]	E[24]	E[24] is the event filter for **IMPLEMENTATION DEFINED** event 24. **0b0** Event 24 is ignored. **0b1** Do not record samples that have event 24 == 1. Access to this field is: **RAZ/WI**	0b0
[23:19]	**RAZ/WI**	Reserved	**RAZ/WI**

Bits	Name	Description	Reset
[18]	E[18]	Not empty predicate. **0b0** Empty predicate event is ignored. **0b1** Do not record samples that have the Empty predicate event == 1.	x
[17]	E[17]	Not partial predicate. **0b0** Partial predicate event is ignored. **0b1** Do not record samples that have the Partial predicate event == 1.	x
[16]	**RAZ/WI**	Reserved	**RAZ/WI**
[15]	E[15]	E[15] is the event filter for **IMPLEMENTATION DEFINED** event 15. **0b0** Event 15 is ignored. **0b1** Do not record samples that have event 15 == 1. Access to this field is: **RAZ/WI**	0b0
[14]	E[14]	E[14] is the event filter for **IMPLEMENTATION DEFINED** event 14. **0b0** Event 14 is ignored. **0b1** Do not record samples that have event 14 == 1. Access to this field is: **RAZ/WI**	0b0
[13]	E[13]	E[13] is the event filter for **IMPLEMENTATION DEFINED** event 13. **0b0** Event 13 is ignored. **0b1** Do not record samples that have event 13 == 1. Access to this field is: **RAZ/WI**	0b0
[12]	E[12]	E[12] is the event filter for **IMPLEMENTATION DEFINED** event 12. **0b0** Event 12 is ignored. **0b1** Do not record samples that have event 12 == 1. Access to this field is: **RAZ/WI**	0b0
[11]	E[11]	Aligned. **0b0** Misalignment event is ignored. **0b1** Do not record samples that have the Misalignment event == 1.	x

Bits	Name	Description	Reset
[10]	E[10]	**When filtering on event 10 is optionally supported && event 10 is implemented** No remote access. **0b0** Remote access event is ignored. **0b1** Do not record samples that have the Remote access event == 1. **Otherwise** RAZ/WI	xxxx
[9]	E[9]	**When filtering on event 9 is optionally supported && event 9 is implemented** Last Level cache hit. **0b0** Last Level cache miss event is ignored. **0b1** Do not record samples that have the Last Level cache miss event == 1. **Otherwise** RAZ/WI	xxxx
[8]	E[8]	**When filtering on event 8 is optionally supported && event 8 is implemented** No Last Level cache access. **0b0** Last Level cache access event is ignored. **0b1** Do not record samples that have the Last Level cache access event == 1. **Otherwise** RAZ/WI	xxxx
[7]	E[7]	Correctly predicted. **0b0** Mispredicted event is ignored. **0b1** Do not record samples that have the Mispredicted event == 1.	x
[6]	E[6]	Taken. **0b0** Not taken event is ignored. **0b1** Do not record samples that have the Not taken event == 1.	x
[5]	E[5]	TLB hit. **0b0** TLB walk event is ignored. **0b1** Do not record samples that have the TLB walk event == 1.	x

Bits	Name	Description	Reset
[4]	E[4]	**When filtering on event 4 is optionally supported** No TLB access. **0b0** TLB access event is ignored. **0b1** Do not record samples that have the TLB access event == 1. **Otherwise** RAZ/WI	xxxx
[3]	E[3]	Level 1 data or unified cache hit. **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 == 1.	x
[2]	E[2]	**When filtering on event 2 is optionally supported** No Level 1 data cache access. **0b0** Level 1 data cache access event is ignored. **0b1** Do not record samples that have the Level 1 data cache access event == 1. **Otherwise** RAZ/WI	xxxx
[1]	E[1]	Speculative. **0b0** Architecturally executed event is ignored. **0b1** Do not record samples that have the Architecturally executed event == 1.	x
[0]	**RAZ/WI**	Reserved	**RAZ/WI**

Access

MRS <Xt>, PMSNEVFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b001

MSR PMSNEVFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b001

Accessibility

MRS <Xt>, PMSNEVFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.nPMSNEVFR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x850];
    else
        X[t, 64] = PMSNEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSNEVFR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSNEVFR_EL1;
```

MSR PMSNEVFR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.nPMSNEVFR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x850] = X[t, 64];
    else
        PMSNEVFR_EL1 = X[t, 64];
```

```
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif MDCR_EL3.EnPMSN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMSNEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMSNEVFR_EL1 = X[t, 64];
```

A.17.2 PMSEVFR_EL1, Sampling Event Filter Register

Controls sample filtering by events. The overall filter is the logical AND of these filters. For example, if PMSEVFR_EL1.E[3] and PMSEVFR_EL1.E[5] are both set to 1, only samples that have both event 3 (Level 1 unified or data cache refill) and event 5 (TLB walk) set to 1 are recorded.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Statistical Profiling Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx 0000 0000 0000 0000 xxxx xxxx 0000 0xx0 xxxx xxxx xxxx xxx0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-282: AArch64_pmsevfr_el1 bit assignments

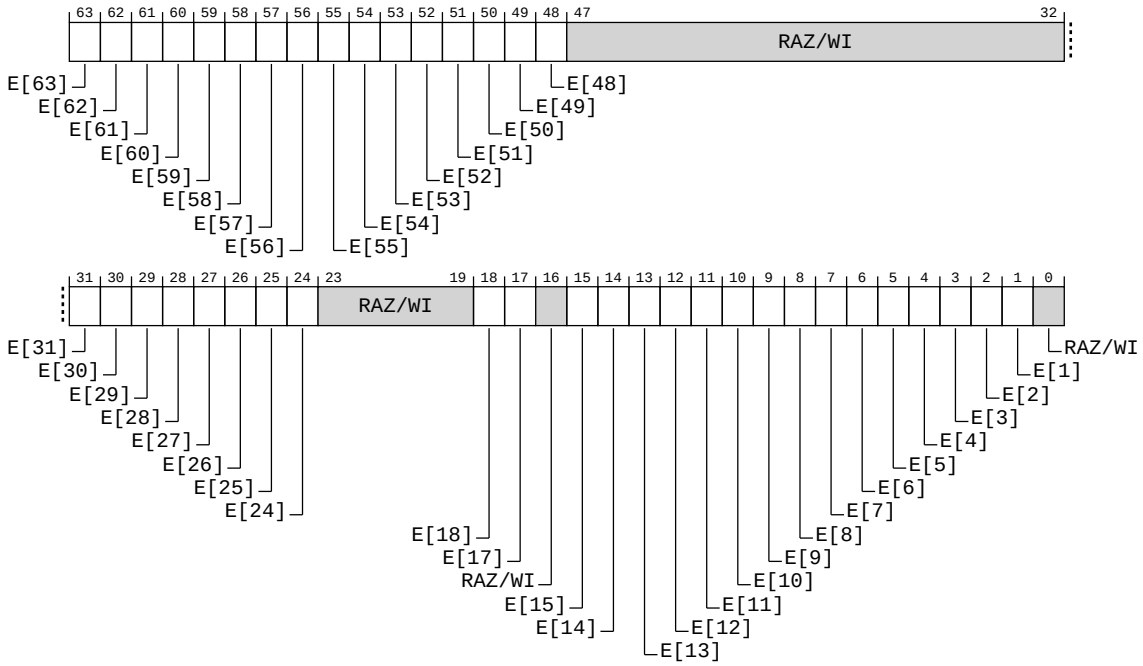


Table A-749: PMSEVFR_EL1 bit descriptions

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.	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.	x
[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.	x

Bits	Name	Description	Reset
[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.	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.	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.	x
[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.	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.	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.	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.	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.	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.	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.	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.	x
[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.	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.	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.	x

Bits	Name	Description	Reset
[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.	x
[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.	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.	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.	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.	x
[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.	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.	x
[23:19]	**RAZ/WI**	Reserved	**RAZ/WI**

Bits	Name	Description	Reset
[18]	E[18]	Empty predicate. **0b0** Empty predicate event is ignored. **0b1** Do not record samples that have the Empty predicate event == 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.	x
[16]	**RAZ/WI**	Reserved	**RAZ/WI**
[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.	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.	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.	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.	x
[11]	E[11]	Alignment. **0b0** Alignment event is ignored. **0b1** Do not record samples that have the Alignment event == 0.	x

Bits	Name	Description	Reset
[10]	E[10]	**When filtering on event 10 is optionally supported && event 10 is implemented** Remote access. **0b0** Remote access event is ignored. **0b1** Do not record samples that have the Remote access event == 0. **Otherwise** RAZ/WI	xxxx
[9]	E[9]	**When filtering on event 9 is optionally supported && event 9 is implemented** Last Level cache miss. **0b0** Last Level cache miss event is ignored. **0b1** Do not record samples that have the Last Level cache miss event == 0. **Otherwise** RAZ/WI	xxxx
[8]	E[8]	**When filtering on event 8 is optionally supported && event 8 is implemented** Last Level cache access. **0b0** Last Level cache access event is ignored. **0b1** Do not record samples that have the Last Level cache access event == 0. **Otherwise** RAZ/WI	xxxx
[7]	E[7]	Mispredicted. **0b0** Mispredicted event is ignored. **0b1** Do not record samples that have the Mispredicted event == 0.	x
[6]	E[6]	Not taken. **0b0** Not taken event is ignored. **0b1** Do not record samples that have the Not taken event == 0.	x
[5]	E[5]	TLB walk. **0b0** TLB walk event is ignored. **0b1** Do not record samples that have the TLB walk event == 0.	x

Bits	Name	Description	Reset
[4]	E[4]	**When filtering on event 4 is optionally supported** TLB access. **0b0** TLB access event is ignored. **0b1** Do not record samples that have the TLB access event == 0. **Otherwise** RAZ/WI	xxxx
[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.	x
[2]	E[2]	**When filtering on event 2 is optionally supported** Level 1 data cache access. **0b0** Level 1 data cache access event is ignored. **0b1** Do not record samples that have the Level 1 data cache access event == 0. **Otherwise** RAZ/WI	xxxx
[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.	x
[0]	**RAZ/WI**	Reserved	**RAZ/WI**

Access

MRS <Xt>, PMSEVFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101

MSR PMSEVFR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b101

Accessibility

MRS <Xt>, PMSEVFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = NVMem[0x830];
    else
        X[t, 64] = PMSEVFR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) 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() && SCR_EL3.FGTEn == '1' && HDFGWTR_EL2.PMSEVFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x830] = X[t, 64];
    else
        PMSEVFR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) 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.17.3 PMSIDR_EL1, Sampling Profiling ID Register

Describes the Statistical Profiling implementation to software

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Statistical Profiling Extension registers

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0010 0110 0100 xx01 0xxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |   | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-283: AArch64_pmsidr_el1 bit assignments

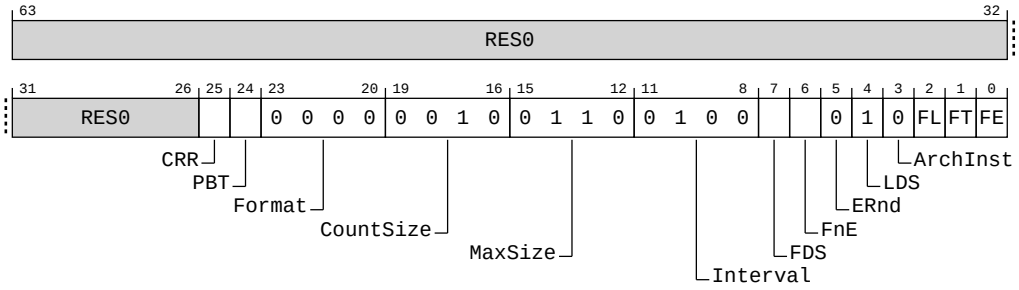


Table A-752: PMSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:26]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[25]	CRR	Call Return branch records. Defined values are: **0b0** Operation Type packets for branches do not contain Call Return information. **0b1** Operation Type packets for branches contain Call Return information.	The reset values can be the following: 0b0, 0b1, respective to the value.
[24]	PBT	Previous branch target Address packet. Defined values are: **0b0** Previous branch target Address packet not supported. **0b1** Previous branch target Address packet support implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[23:20]	Format	Defines the format of the sample records. Defined values are: **0b0000** Format 0.	0b0000
[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]	FDS	Filter by data source. Defined values are: **0b0** AArch64-PMSDSFR_EL1 is not implemented and AArch64-PMSFCR_EL1.FDS is **RES0**. **0b1** AArch64-PMSDSFR_EL1 and AArch64-PMSFCR_EL1.FDS are implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[6]	FnE	Filtering by events, inverted. Defined values are: **0b0** AArch64-PMSNEVFR_EL1 is not implemented and AArch64-PMSFCR_EL1.FnE is **RES0**. **0b1** AArch64-PMSNEVFR_EL1 and AArch64-PMSFCR_EL1.FnE are implemented.	The reset values can be the following: 0b0, 0b1, respective to the value.
[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

Bits	Name	Description	Reset
[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() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMSIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.NSPB[0] == '0' || MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) 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 ||
 (IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE != SCR_EL3.NSE) 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.17.4 PMBIDR_EL1, Profiling Buffer ID Register

Provides information to software as to whether the buffer can be programmed at the current Exception level.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-284: AArch64_pmbidr_el1 bit assignments

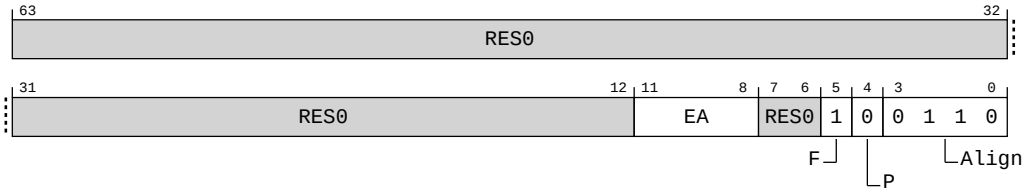


Table A-754: PMBIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[11:8]	EA	External Abort handling. Describes how the PE manages External aborts on writes made by the Statistical Profiling Unit to the Profiling Buffer. **0b0000** Not described. **0b0001** The PE ignores External aborts on writes made by the Statistical Profiling Unit. **0b0010** The External abort generates an SError interrupt at the PE.	The reset values can be the following: 0b0000, 0b0001, 0b0010, respective to the value.
[7:6]	**RES0**	Reserved	**RES0**
[5]	F	Flag updates. Describes how address translations performed by the Statistical Profiling Unit manage the Access flag and dirty state. **0b1** Hardware management of the Access flag and dirty state for accesses made by the Statistical Profiling Unit is controlled in the same way as explicit memory accesses in the Profiling Buffer owning translation regime.	0b1
[4]	P	Programming not allowed. When read at EL3, this field reads as zero. Otherwise, indicates that the Profiling Buffer is owned by a higher Exception level or another Security state. 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
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMBIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMBIDR_EL1;
```

A.18 AArch64 Trace Buffer Extension registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table A-756: Trace Buffer Extension registers summary

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

Appendix B External registers

This appendix contains the descriptions for the Neoverse™ V3 memory mapped 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 following summary table provides an overview of all memory-mapped CoreROM registers in the core.

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

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

Table B-1: CoreROM registers summary

Offset	Name	Reset	Width	Description
0x000	COREROM_ROMENTRY0	See individual bit resets.	32-bit	Core ROM table Entry 0
0x004	COREROM_ROMENTRY1	See individual bit resets.	32-bit	Core ROM table Entry 1
0x008	COREROM_ROMENTRY2	See individual bit resets.	32-bit	Core ROM table Entry 2
0x00C	COREROM_ROMENTRY3	See individual bit resets.	32-bit	Core ROM table Entry 3
0xFB8	COREROM_AUTHSTATUS	See individual bit resets.	32-bit	Core ROM table Authentication Status Register
0xFBC	COREROM_DEVARCH	See individual bit resets.	32-bit	Core ROM table Device Architecture Register
0xFCC	COREROM_DEVTYPE	See individual bit resets.	32-bit	Core ROM table Device Type Register
0xFD0	COREROM_PIDR4	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 4
0xFE0	COREROM_PIDR0	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 0
0xFE4	COREROM_PIDR1	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 1
0xFE8	COREROM_PIDR2	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 2
0xFEC	COREROM_PIDR3	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 3
0xFF0	COREROM_CIDR0	See individual bit resets.	32-bit	Core ROM table Component Identification Register 0
0xFF4	COREROM_CIDR1	See individual bit resets.	32-bit	Core ROM table Component Identification Register 1
0xFF8	COREROM_CIDR2	See individual bit resets.	32-bit	Core ROM table Component Identification Register 2
0xFFC	COREROM_CIDR3	See individual bit resets.	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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-1: ext_corerom_romentry0 bit assignments

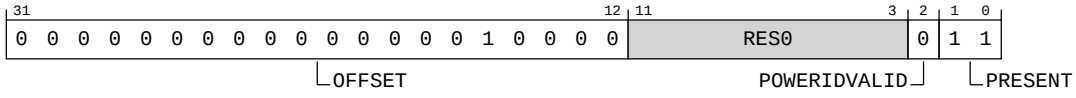


Table B-2: COREROM_ROMENTRY0 bit descriptions

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**

Bits	Name	Description	Reset
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0b0** A power domain ID is not provided.	0b0
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **0b11** The ROM Entry is present.	0b11

Accessibility

Component	Offset	Range
CoreROM	0x000	None

This interface is accessible as follows:

RO

B.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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-2: ext_corerom_romentry1 bit assignments

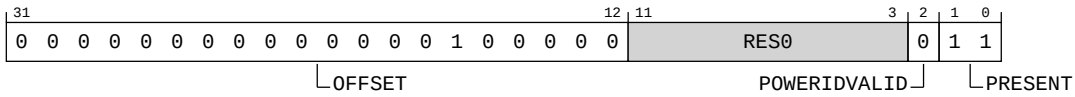


Table B-4: COREROM_ROMENTRY1 bit descriptions

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

Component	Offset	Range
CoreROM	0x004	None

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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-3: ext_corerom_romentry2 bit assignments

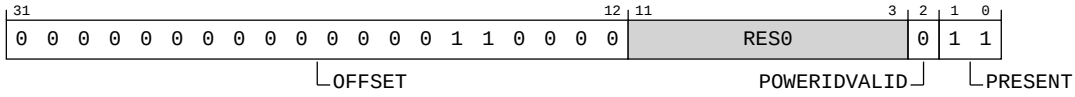


Table B-6: COREROM_ROMENTRY2 bit descriptions

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

Component	Offset	Range
CoreROM	0x008	None

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-4: ext_corerom_romentry3 bit assignments

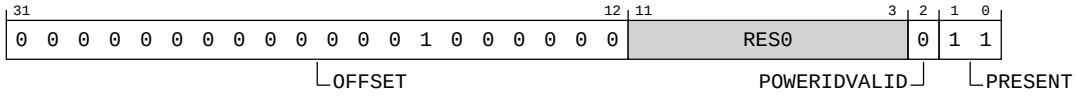


Table B-8: COREROM_ROMENTRY3 bit descriptions

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

Component	Offset	Range
CoreROM	0x00C	None

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 xxxx 0000
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-5: ext_corerom_authstatus bit assignments

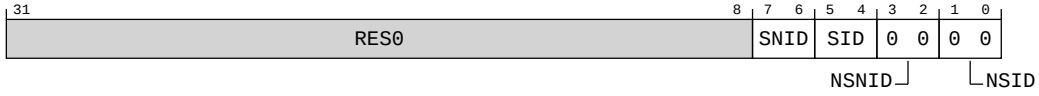


Table B-10: COREROM_AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:6]	SNID	Secure Non-invasive Debug.	xx
[5:4]	SID	Secure Invasive Debug.	xx
[3:2]	NSNID	Non-secure Non-invasive Debug. **0b00** Debug level is not supported.	0b00
[1:0]	NSID	Non-secure Invasive Debug. **0b00** Debug level is not supported.	0b00

Accessibility

Component	Offset	Range
CoreROM	0xFB8	None

This interface is accessible as follows:

RO

B.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

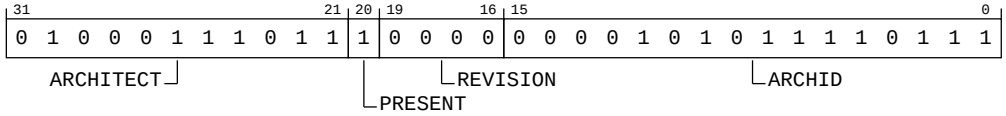


Table B-12: COREROM_DEVARCH bit descriptions

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

Component	Offset	Range
CoreROM	0xFBC	None

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-7: ext_corerom_devtype bit assignments

Table B-14: COREROM_DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	SUB	Sub number **0b0000** Other, undefined.	0b0000
[3:0]	MAJOR	Major number **0b0000** Miscellaneous.	0b0000

Accessibility

Component	Offset	Range
CoreROM	0xFCC	None

This interface is accessible as follows:

RO

B.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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-8: ext_corerom_pidr4 bit assignments

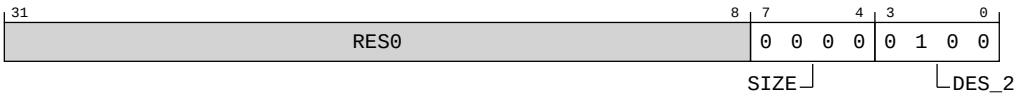


Table B-16: COREROM_PIDR4 bit descriptions

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

Component	Offset	Range
CoreROM	0xFD0	None

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 1000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-9: ext_corerom_pidr0 bit assignments

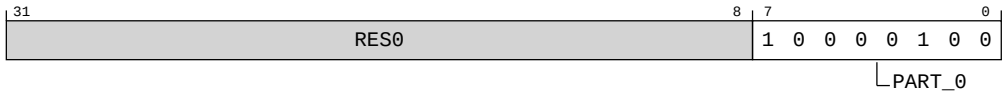


Table B-18: COREROM_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number bits [7:0]. **0b10000100** Neoverse V3 Core ROM table. Bits [7:0] of part number 0xD84.	0x84

Accessibility

Component	Offset	Range
CoreROM	0xFE0	None

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-10: ext_corerom_pidr1 bit assignments

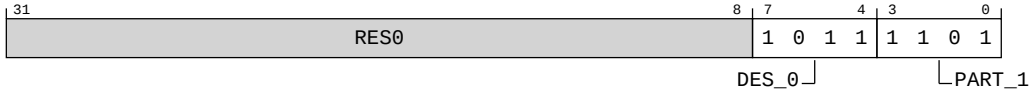


Table B-20: COREROM_PIDR1 bit descriptions

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
[3:0]	PART_1	Part number bits [11:8]. **0b1101** Neoverse V3 Core ROM table. Bits [11:8] of part number 0xD84.	0b1101

Accessibility

Component	Offset	Range
CoreROM	0xFE4	None

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 0000 1011
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-11: ext_corerom_pidr2 bit assignments

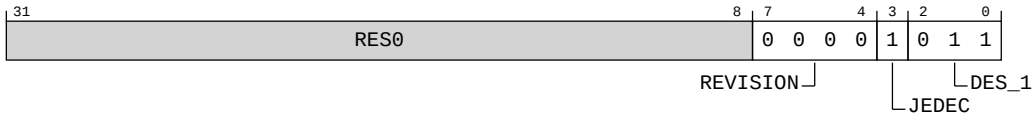


Table B-22: COREROM_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	REVISION	Component revision. **0b0000** r0p1	0b0000
[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

Component	Offset	Range
CoreROM	0xFE8	None

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 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-12: ext_corerom_pidr3 bit assignments

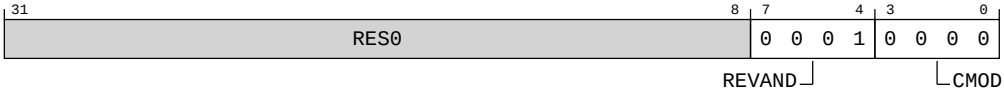


Table B-24: COREROM_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Minor errata fixes. **0b0001**	0b0001
[3:0]	CMOD	Customer Modified. **0b0000** The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
CoreROM	0xFEC	None

This interface is accessible as follows:

RO

B.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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-13: ext_corerom_cidr0 bit assignments

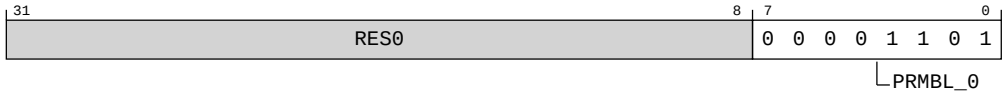


Table B-26: COREROM_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	CoreSight component identification preamble. **0b00001101** CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Range
CoreROM	0xFF0	None

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-14: ext_corerom_cidr1 bit assignments

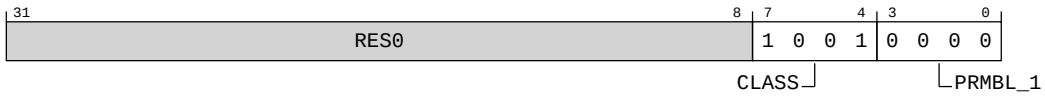


Table B-28: COREROM_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	CLASS	CoreSight component class. **0b1001** CoreSight component.	0b1001

Bits	Name	Description	Reset
[3:0]	PRMBL_1	CoreSight component identification preamble. **0b0000** CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
CoreROM	0xFF4	None

This interface is accessible as follows:

RO

B.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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-15: ext_corerom_cidr2 bit assignments

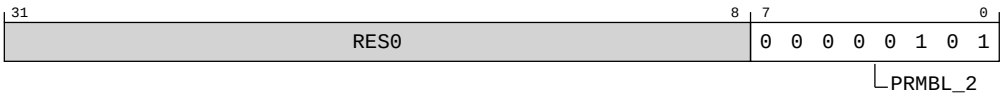


Table B-30: COREROM_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_2	CoreSight component identification preamble. **0b00000101** CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
CoreROM	0xFF8	None

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
```

```
|    |    |    |    |    |    |    |    |
31   27   23   19   15   11   7    3   0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-16: ext_corerom_cidr3 bit assignments

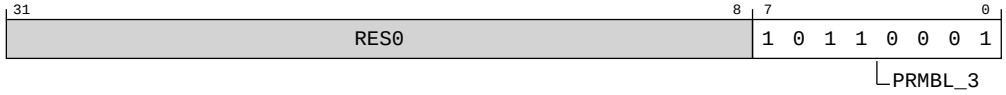


Table B-32: COREROM_CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	CoreSight component identification preamble. **0b10110001** CoreSight component identification preamble.	0xB1

Accessibility

Component	Offset	Range
CoreROM	0xFFC	None

This interface is accessible as follows:

RO

B.2 External PPM registers summary

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

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

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

Table B-34: PPM registers summary

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	See individual bit resets.	64-bit	Global Performance and Power Management Configuration Register
0x010	CPUMPMMCR	See individual bit resets.	64-bit	Global MPMM Control Register
0x020	CPUPPMPDPCR	See individual bit resets.	64-bit	Performance and Power Management PDP Control Register
0x080	CPUPPMCR4	See individual bit resets.	64-bit	Power Performance Management Register
0x088	CPUPPMCR5	See individual bit resets.	64-bit	Power Performance Management Register
0x090	CPUPPMCR6	See individual bit resets.	64-bit	Power Performance Management Register

B.2.1 CPUPPMCR, Global Performance and Power Management Configuration Register

Provides **IMPLEMENTATION DEFINED** control and discovery of the Performance and Power Management (PPM) features.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x000

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x111 xxxx x011 xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-17: ext_cpuppmcr bit assignments

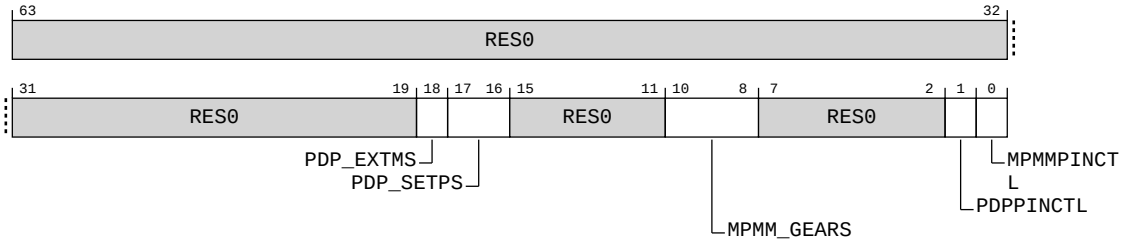


Table B-35: CPUPPMCR bit descriptions

Bits	Name	Description	Reset
[63:19]	**RES0**	Reserved	**RES0**
[18]	PDP_EXTMS	External memory system PDP control **0b1** Independent external memory system PDP control is implemented	0b1
[17:16]	PDP_SETPS	Number of PDP Setpoints Implemented **0b11** 3 PDP setpoints are implemented	0b11
[15:11]	**RES0**	Reserved	**RES0**
[10:8]	MPMM_GEARS	Number of MPMM Gears Implemented **0b011** 3 MPMM gears are implemented	0b011
[7:2]	**RES0**	Reserved	**RES0**
[1]	PDPPINCTL	PDP Pin Control Enabled **0b0** PDP control through SPR and utility bus **0b1** PDP control through pin only	0b0
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0b0** MPMM control through SPR and utility bus **0b1** MPMM control through pin only	0b0

Accessibility

Component	Offset	Range
PPM	0x000	None

This interface is accessible as follows:

RO

B.2.2 CPUMPMMCR, Global MPMM Control Register

Provides **IMPLEMENTATION DEFINED** control of the Maximum Power Mitigation Mechanism (MPMM) feature.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x010

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-18: ext_cpumpmmcr bit assignments

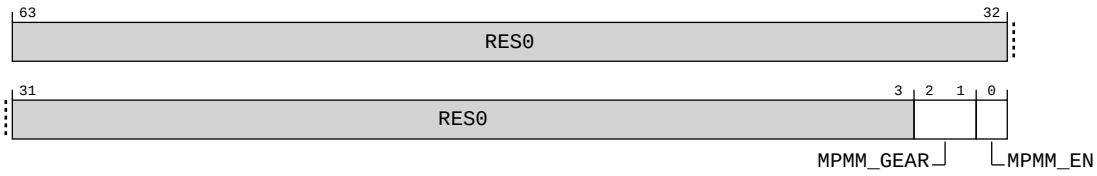


Table B-37: CPUMPMMCR bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2:1]	MPMM_GEAR	MPMM Gear Select **0b00** Select MPMM Gear 0 **0b01** Select MPMM Gear 1 **0b10** Select MPMM Gear 2 **0b11** Select MPMM Gear 3	0b00
[0]	MPMM_EN	MPMM Master Enable **0b0** MPMM is disabled **0b1** MPMM is enabled	0b0

Accessibility

Component	Offset	Range
PPM	0x010	None

This interface is accessible as follows:

RO

B.2.3 CPUPPMPDPCR, Performance and Power Management PDP Control Register

Provides **IMPLEMENTATION DEFINED** control of the Performance Defined Power (PDP) feature.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PPM

Register offset

0x020

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-19: ext_cpuppmpdpcr bit assignments

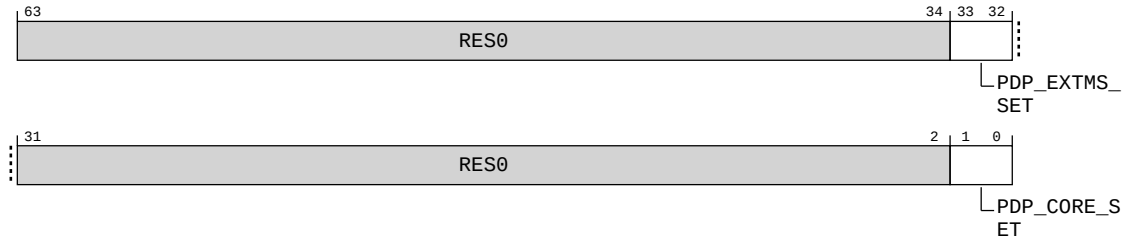


Table B-39: CPUPPMPDPCR bit descriptions

Bits	Name	Description	Reset
[63:34]	**RES0**	Reserved	**RES0**
[33:32]	PDP_EXTMS_SET	External memory system PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00
[31:2]	**RES0**	Reserved	**RES0**
[1:0]	PDP_CORE_SET	Core PDP Aggressiveness **0b00** Disable PDP **0b01** Enable PDP at low aggressiveness **0b10** Enable PDP at medium aggressiveness **0b11** Enable PDP at high aggressiveness	0b00

Accessibility

Component	Offset	Range
PPM	0x020	None

This interface is accessible as follows:

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-20: ext_cpuppmcr4 bit assignments

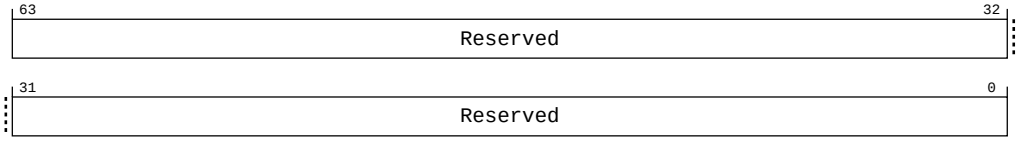


Table B-41: CPUPPMCR4 bit descriptions

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

Accessibility

Component	Offset	Range
PPM	0x080	None

This interface is accessible as follows:

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Reserved

Figure B-21: ext_cpuppmcr5 bit assignments

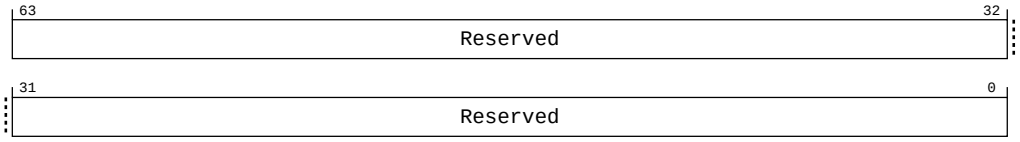


Table B-43: CPUPPMCR5 bit descriptions

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

Accessibility

Component	Offset	Range
PPM	0x088	None

This interface is accessible as follows:

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Reserved

Figure B-22: ext_cpuppmcr6 bit assignments

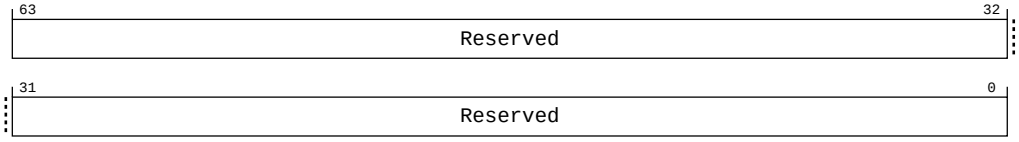


Table B-45: CPUPPMCR6 bit descriptions

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

Accessibility

Component	Offset	Range
PPM	0x090	None

This interface is accessible as follows:

RO

B.3 External PMU registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table B-47: PMU registers summary

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

Offset	Name	Reset	Width	Description
0x20C	PMVIDSR	See individual bit resets.	32-bit	VMID Sample Register
0x22C	PMCID2SR	See individual bit resets.	32-bit	CONTEXTIDR_EL2 Sample Register
0x400	PMEVTYPER0_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x404	PMEVTYPER1_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x408	PMEVTYPER2_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x40C	PMEVTYPER3_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x410	PMEVTYPER4_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x414	PMEVTYPER5_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x418	PMEVTYPER6_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x41C	PMEVTYPER7_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x420	PMEVTYPER8_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x424	PMEVTYPER9_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x428	PMEVTYPER10_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x42C	PMEVTYPER11_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x430	PMEVTYPER12_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x434	PMEVTYPER13_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x438	PMEVTYPER14_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x43C	PMEVTYPER15_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x440	PMEVTYPER16_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x444	PMEVTYPER17_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x448	PMEVTYPER18_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x44C	PMEVTYPER19_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x450	PMEVTYPER20_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x454	PMEVTYPER21_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x458	PMEVTYPER22_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x45C	PMEVTYPER23_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x460	PMEVTYPER24_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x464	PMEVTYPER25_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x468	PMEVTYPER26_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x46C	PMEVTYPER27_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x470	PMEVTYPER28_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x474	PMEVTYPER29_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x478	PMEVTYPER30_EL0	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter Filter Register
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	See individual bit resets.	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register

Offset	Name	Reset	Width	Description
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x668	PMEVCNTSR9	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C0	PMEVCNTSR20	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6C8	PMEVCNTSR21	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D0	PMEVCNTSR22	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6D8	PMEVCNTSR23	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E0	PMEVCNTSR24	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6E8	PMEVCNTSR25	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMEVCNTSR26	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F8	PMEVCNTSR27	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x700	PMEVCNTSR28	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x708	PMEVCNTSR29	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x710	PMEVCNTSR30	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0xC00	PMCNTENSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Set register
0xC20	PMCNTENCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Clear register
0xC40	PMINTENSET_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Set register
0xC60	PMINTENCLR_EL1 [31:0]	See individual bit resets.	32-bit	Performance Monitors Interrupt Enable Clear register
0xC80	PMOVSCLR_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCA0	PMSWINC_EL0	See individual bit resets.	32-bit	Performance Monitors Software Increment register
0xCC0	PMOVSSET_EL0 [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Set register
0xE00	PMCFGR [31:0]	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_EL0	See individual bit resets.	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 0

Offset	Name	Reset	Width	Description
0xE24	PMCEID1	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE30	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE40	PMMIR [31:0]	See individual bit resets.	32-bit	Performance Monitors Machine Identification Register
0xFA8	PMDEVAFF0	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	See individual bit resets.	32-bit	Performance Monitors Device Affinity register 1
0xFB0	PMLAR	See individual bit resets.	32-bit	Performance Monitors Lock Access Register
0xFB4	PMLSR	See individual bit resets.	32-bit	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	See individual bit resets.	32-bit	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	See individual bit resets.	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	See individual bit resets.	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	See individual bit resets.	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 3

B.3.1 PMPCSSR, Snapshot Program Counter Sample Register

Captured copy of the Program Counter.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x600

Access type

Read

R

Write

RESERVED

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-23: ext_pmpcssr bit assignments

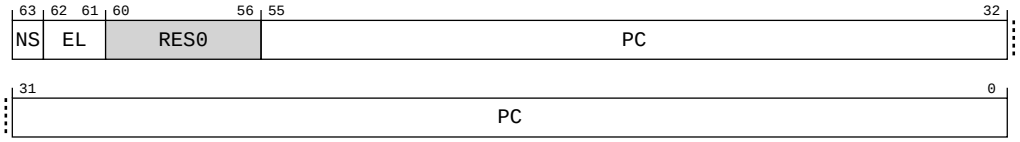


Table B-48: PMPCSSR bit descriptions

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.	56{x}

Accessibility

PMPCSSR is set to an UNKNOWN value by a read of the EDPCSRlo or PMPCSR[31:0] register.

Component	Offset	Range
PMU	0x600	None

This interface is accessible as follows:

RO

B.3.2 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register

Captured copy of the CONTEXTIDR_EL1 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x608

Access type

Read

R

Write

RESERVED

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-24: ext_pmcidssr bit assignments

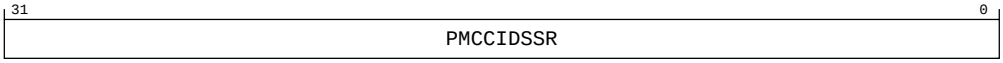


Table B-50: PMCIDSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCIDSSR	PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot.	32{x}

Accessibility

PMCIDSSR is set to an UNKNOWN value by a read of the EDPCSRlo or PMPCSR[31:0] register.

Component	Offset	Range
PMU	0x608	None

This interface is accessible as follows:

RO

B.3.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register

Captured copy of the CONTEXTIDR_EL2 register.

The value captured must relate to the instruction captured in PMPCSSR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x60C

Access type

Read

R

Write

RESERVED

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-25: ext_pmcid2ssr bit assignments

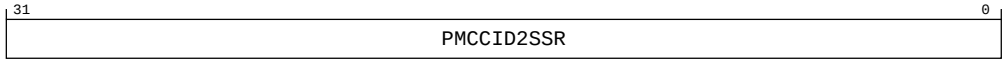


Table B-52: PMCID2SSR bit descriptions

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.

Component	Offset	Range
PMU	0x60C	None

This interface is accessible as follows:

RO

B.3.4 PMSSSR, PMU Snapshot Status Register

Holds status information about the captured counters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PMU

Register offset

0x610

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-26: ext_pmsssr bit assignments

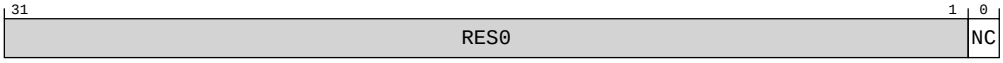


Table B-54: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	**RES0**	Reserved	**RES0**
[0]	NC	No capture. Indicates whether the PMU counters have been captured. **0b0** PMU counters captured. **0b1** PMU counters not captured.	x

Accessibility

Component	Offset	Range
PMU	0x610	None

This interface is accessible as follows:

RO

B.3.5 PMCCNTSR, PMU Cycle Counter Snapshot Register

Captured copy of PMCCNTR_EL0. Once captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR_EL0 and PMCR_EL0.C.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x618

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-27: ext_pmccntsr bit assignments

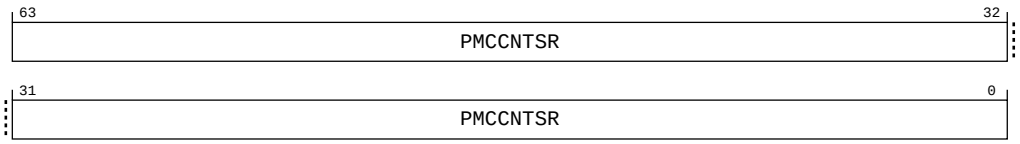


Table B-56: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_EL0 sample. Sampled cycle count.	64{x}

Accessibility

Component	Offset	Range
PMU	0x618	None

This interface is accessible as follows:

RO

B.3.6 PMEVCNTSR0, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x620

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-28: ext_pmevcntsr0 bit assignments

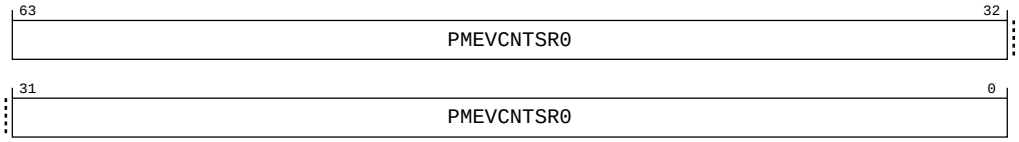


Table B-58: PMEVCNTSR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR0	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x620	PMEVCNTSR0	None

This interface is accessible as follows:

RO

B.3.7 PMEVCNTSR1, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x628

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-29: ext_pmevcntsr1 bit assignments

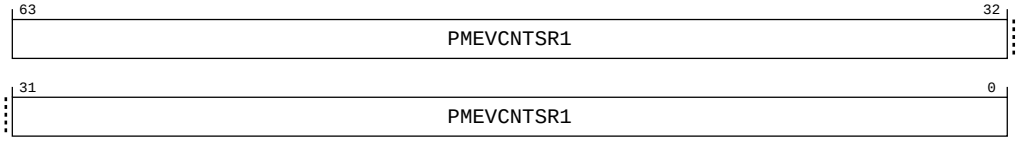


Table B-60: PMEVCNTSR1 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR1	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x628	PMEVCNTSR1	None

This interface is accessible as follows:

RO

B.3.8 PMEVCNTSR2, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x630

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-30: ext_pmevcntsr2 bit assignments

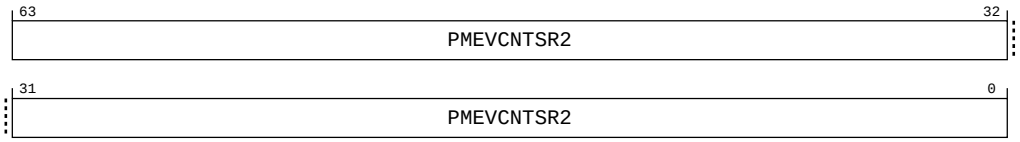


Table B-62: PMEVCNTSR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR2	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x630	PMEVCNTSR2	None

This interface is accessible as follows:

RO

B.3.9 PMEVCNTSR3, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x638

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-31: ext_pmevcntsr3 bit assignments

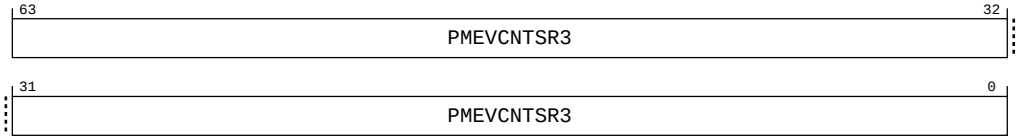


Table B-64: PMEVCNTSR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR3	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x638	PMEVCNTSR3	None

This interface is accessible as follows:

RO

B.3.10 PMEVCNTSR4, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x640

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-32: ext_pmevcntsr4 bit assignments

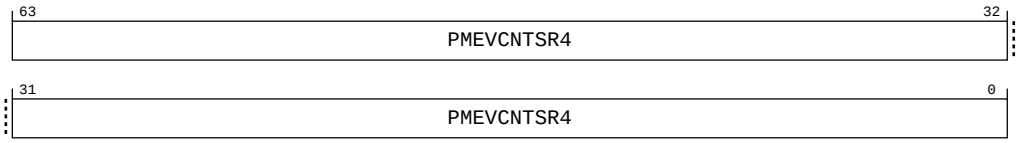


Table B-66: PMEVCNTSR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR4	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x640	PMEVCNTSR4	None

This interface is accessible as follows:

RO

B.3.11 PMEVCNTSR5, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x648

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-33: ext_pmevcntsr5 bit assignments

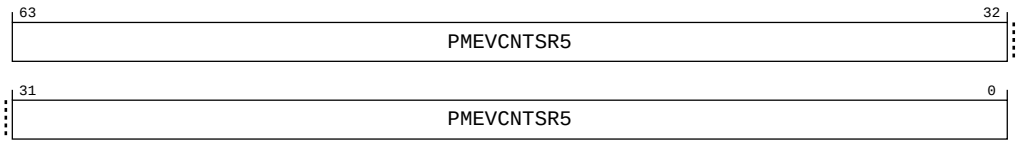


Table B-68: PMEVCNTSR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR5	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x648	PMEVCNTSR5	None

This interface is accessible as follows:

RO

B.3.12 PMEVCNTSR6, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x650

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-34: ext_pmevcntsr6 bit assignments

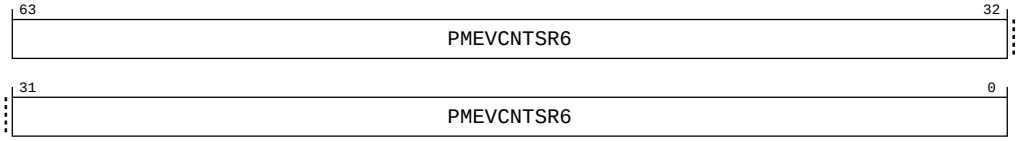


Table B-70: PMEVCNTSR6 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR6	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x650	PMEVCNTSR6	None

This interface is accessible as follows:

RO

B.3.13 PMEVCNTSR7, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x658

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-35: ext_pmevcntsr7 bit assignments

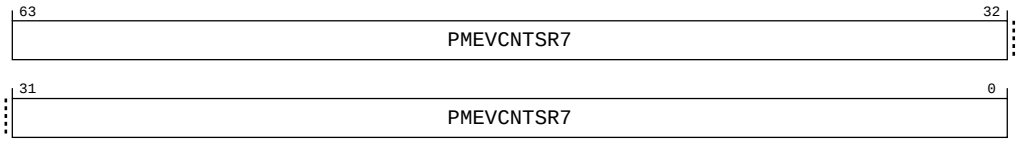


Table B-72: PMEVCNTSR7 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR7	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x658	PMEVCNTSR7	None

This interface is accessible as follows:

RO

B.3.14 PMEVCNTSR8, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x660

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-36: ext_pmevcntsr8 bit assignments

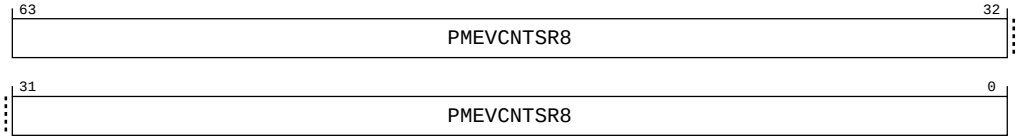


Table B-74: PMEVCNTSR8 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR8	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x660	PMEVCNTSR8	None

This interface is accessible as follows:

RO

B.3.15 PMEVCNTSR9, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x668

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-37: ext_pmevcntsr9 bit assignments

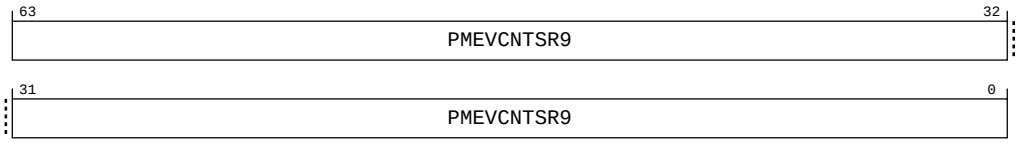


Table B-76: PMEVCNTSR9 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR9	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x668	PMEVCNTSR9	None

This interface is accessible as follows:

RO

B.3.16 PMEVCNTSR10, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x670

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-38: ext_pmevcntsr10 bit assignments

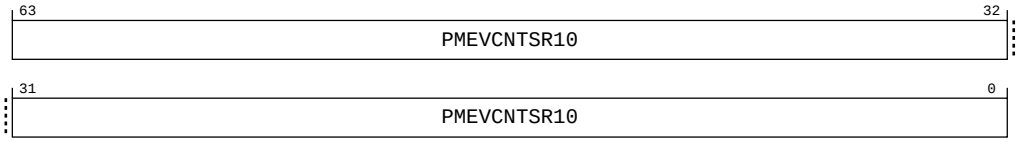


Table B-78: PMEVCNTSR10 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR10	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x670	PMEVCNTSR10	None

This interface is accessible as follows:

RO

B.3.17 PMEVCNTSR11, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x678

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-39: ext_pmevcntsr11 bit assignments

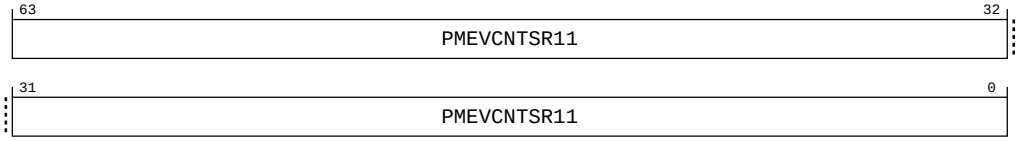


Table B-80: PMEVCNTSR11 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR11	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x678	PMEVCNTSR11	None

This interface is accessible as follows:

RO

B.3.18 PMEVCNTSR12, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x680

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-40: ext_pmevcntsr12 bit assignments

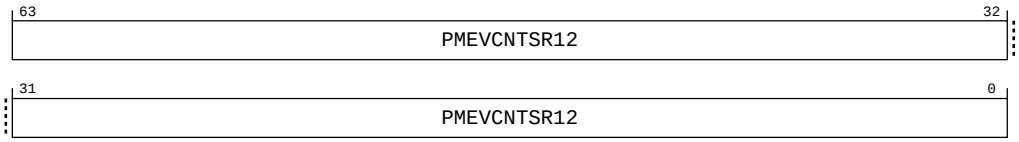


Table B-82: PMEVCNTSR12 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR12	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x680	PMEVCNTSR12	None

This interface is accessible as follows:

RO

B.3.19 PMEVCNTSR13, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x688

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-41: ext_pmevcntsr13 bit assignments

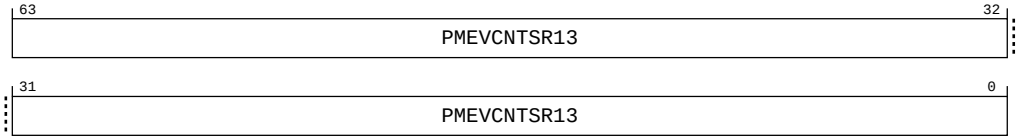


Table B-84: PMEVCNTSR13 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR13	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x688	PMEVCNTSR13	None

This interface is accessible as follows:

RO

B.3.20 PMEVCNTSR14, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x690

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-42: ext_pmevcntsr14 bit assignments

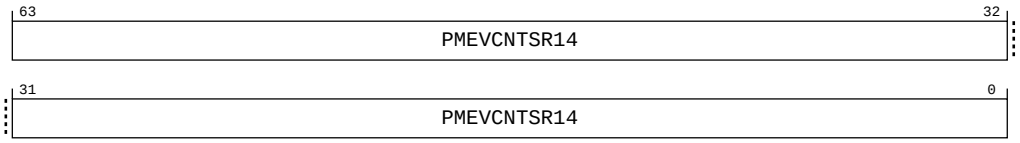


Table B-86: PMEVCNTSR14 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR14	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x690	PMEVCNTSR14	None

This interface is accessible as follows:

RO

B.3.21 PMEVCNTSR15, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x698

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-43: ext_pmevcntsr15 bit assignments

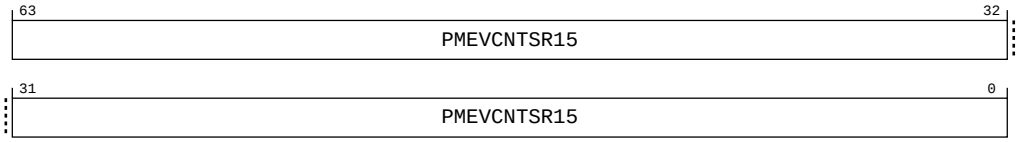


Table B-88: PMEVCNTSR15 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR15	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x698	PMEVCNTSR15	None

This interface is accessible as follows:

RO

B.3.22 PMEVCNTSR16, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6A0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-44: ext_pmevcntsr16 bit assignments

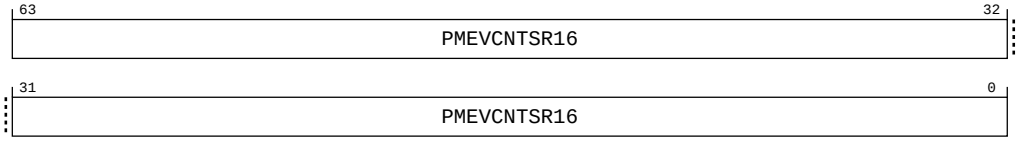


Table B-90: PMEVCNTSR16 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR16	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6A0	PMEVCNTSR16	None

This interface is accessible as follows:

RO

B.3.23 PMEVCNTSR17, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6A8

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-45: ext_pmevcntsr17 bit assignments

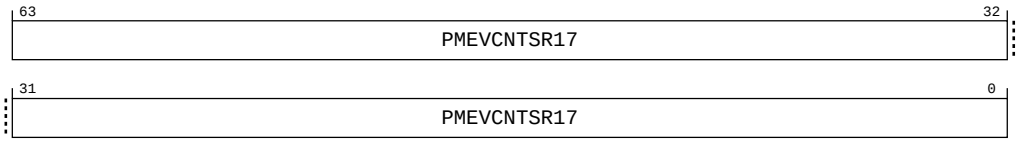


Table B-92: PMEVCNTSR17 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR17	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6A8	PMEVCNTSR17	None

This interface is accessible as follows:

RO

B.3.24 PMEVCNTSR18, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6B0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-46: ext_pmevcntsr18 bit assignments

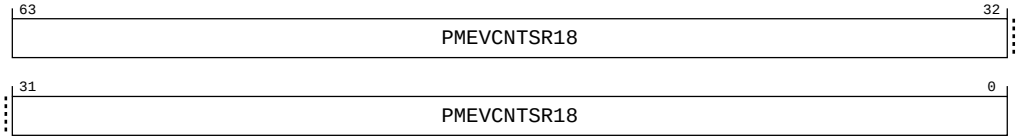


Table B-94: PMEVCNTSR18 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR18	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6B0	PMEVCNTSR18	None

This interface is accessible as follows:

RO

B.3.25 PMEVCNTSR19, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6B8

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-47: ext_pmevcntsr19 bit assignments

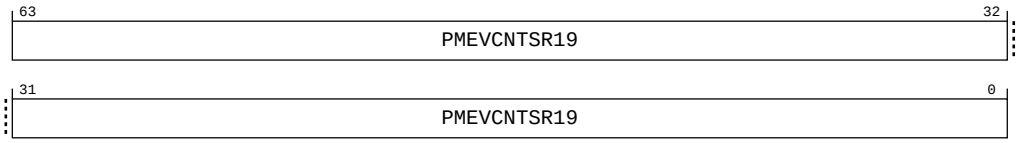


Table B-96: PMEVCNTSR19 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR19	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6B8	PMEVCNTSR19	None

This interface is accessible as follows:

RO

B.3.26 PMEVCNTSR20, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6C0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-48: ext_pmevcntsr20 bit assignments

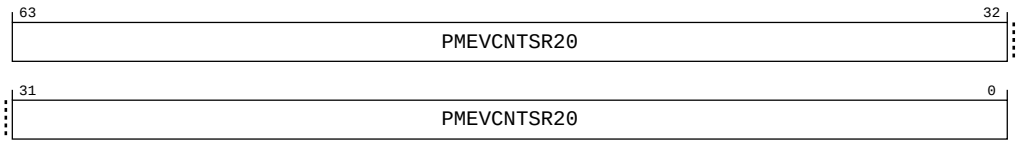


Table B-98: PMEVCNTSR20 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR20	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6C0	PMEVCNTSR20	None

This interface is accessible as follows:

RO

B.3.27 PMEVCNTSR21, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6C8

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-49: ext_pmevcntsr21 bit assignments

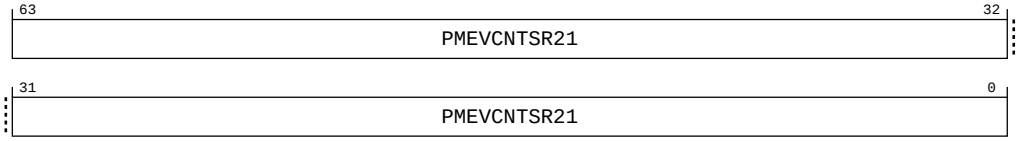


Table B-100: PMEVCNTSR21 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR21	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6C8	PMEVCNTSR21	None

This interface is accessible as follows:

RO

B.3.28 PMEVCNTSR22, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6D0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-50: ext_pmevcntsr22 bit assignments

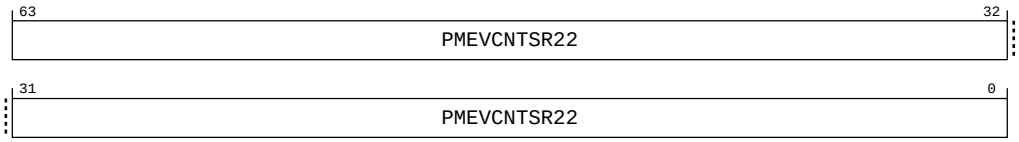


Table B-102: PMEVCNTSR22 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR22	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6D0	PMEVCNTSR22	None

This interface is accessible as follows:

RO

B.3.29 PMEVCNTSR23, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6D8

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-51: ext_pmevcntsr23 bit assignments

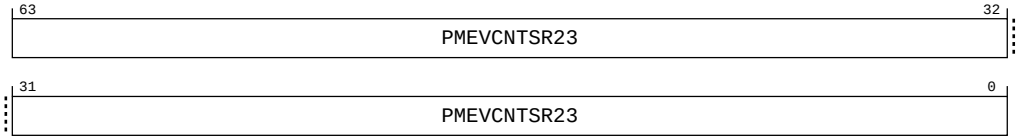


Table B-104: PMEVCNTSR23 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR23	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6D8	PMEVCNTSR23	None

This interface is accessible as follows:

RO

B.3.30 PMEVCNTSR24, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6E0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-52: ext_pmevcntsr24 bit assignments

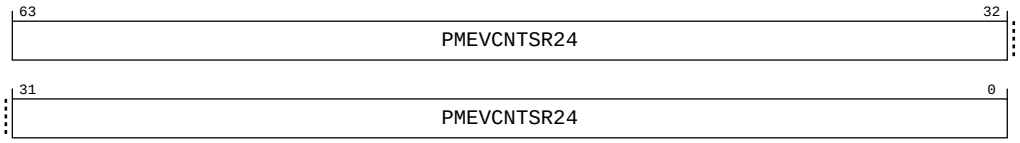


Table B-106: PMEVCNTSR24 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR24	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6E0	PMEVCNTSR24	None

This interface is accessible as follows:

RO

B.3.31 PMEVCNTSR25, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6E8

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-53: ext_pmevcntsr25 bit assignments

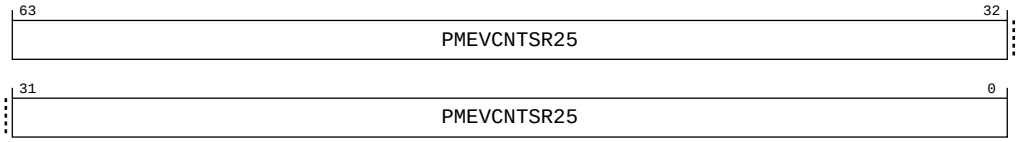


Table B-108: PMEVCNTSR25 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR25	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6E8	PMEVCNTSR25	None

This interface is accessible as follows:

RO

B.3.32 PMEVCNTSR26, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6F0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-54: ext_pmevcntsr26 bit assignments

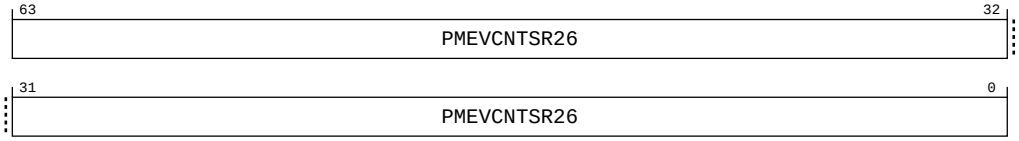


Table B-110: PMEVCNTSR26 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR26	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6F0	PMEVCNTSR26	None

This interface is accessible as follows:

RO

B.3.33 PMEVCNTSR27, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x6F8

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-55: ext_pmevcntsr27 bit assignments

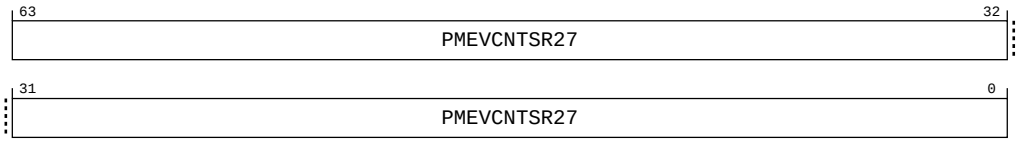


Table B-112: PMEVCNTSR27 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR27	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x6F8	PMEVCNTSR27	None

This interface is accessible as follows:

RO

B.3.34 PMEVCNTSR28, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x700

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-56: ext_pmevcntsr28 bit assignments

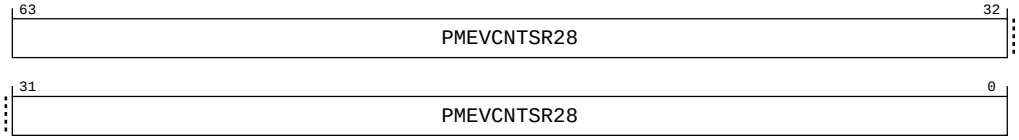


Table B-114: PMEVCNTSR28 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR28	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x700	PMEVCNTSR28	None

This interface is accessible as follows:

RO

B.3.35 PMEVCNTSR29, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x708

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-57: ext_pmevcntsr29 bit assignments

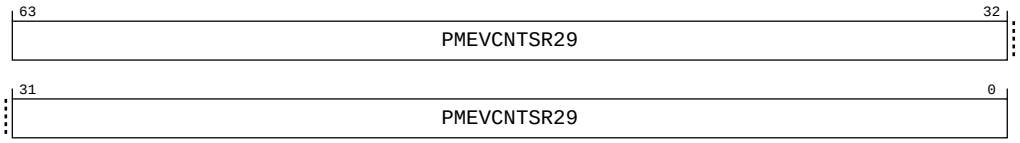


Table B-116: PMEVCNTSR29 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR29	PMEVCNTR<n>_EL0 sample. Sampled event count.	64{x}

Accessibility

Component	Offset	Instance	Range
PMU	0x708	PMEVCNTSR29	None

This interface is accessible as follows:

RO

B.3.36 PMEVCNTSR30, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.P.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offset

0x710

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-58: ext_pmevcntsr30 bit assignments

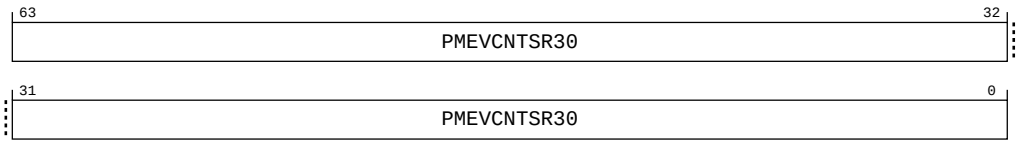


Table B-118: PMEVCNTSR30 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR30	PMEVCNTR<n>_EL0 sample. Sampled event count.	`64{x}`

Accessibility

Component	Offset	Instance	Range
PMU	0x710	PMEVCNTSR30	None

This interface is accessible as follows:

RO

B.3.37 PMCFGR, Performance Monitors Configuration Register

Contains PMU-specific configuration data.

Configurations

PMCFGR is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xE00

Access type

See bit descriptions

Reset value

```
0000 xxxx xx1x 0000 0111 1111 0000 0110
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-59: pmu_pmcfgr bit assignments

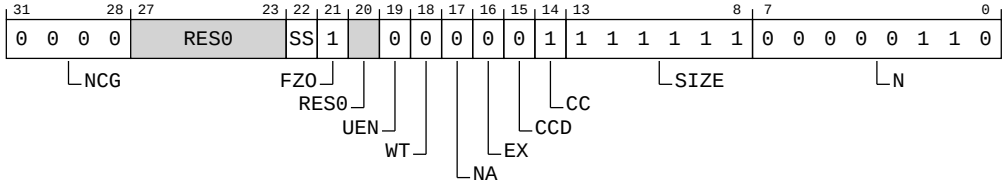


Table B-120: PMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	Defines the number of counter groups implemented, minus one. This field reads-as-zero. **0b0000**	0b0000
[27:23]	**RES0**	Reserved	**RES0**
[22]	SS	Snapshot supported. **0b0** Snapshot mechanism not supported. The locations 0x600-0x7FC and 0xE30-0xE3C are **IMPLEMENTATION DEFINED**. **0b1** Snapshot mechanism supported. Locations 0x600-0x7FC and 0xE30-0xE3C contain **IMPLEMENTATION DEFINED** snapshot registers.	The reset values can be the following: 0b0, 0b1, respective to the value.
[21]	FZO	Freeze-on-overflow supported. Defined values are: **0b1** Freeze-on-overflow mechanism is supported. PMU.PMCR_EL0.FZO is RW.	0b1
[20]	**RES0**	Reserved	**RES0**
[19]	UEN	User-mode Enable Register supported. AArch64-PMUSERENR_EL0 is not visible in the external debug interface, so this bit is **RAZ**. **0b0**	0b0
[18]	WT	This feature is not supported, so this bit is **RAZ**. **0b0**	0b0
[17]	NA	This feature is not supported, so this bit is **RAZ**. **0b0**	0b0

Bits	Name	Description	Reset
[16]	EX	Export supported. **0b0** PMU.PMCR_EL0.X is **RES0**.	0b0
[15]	CCD	Cycle counter has prescale. **0b0** PMU.PMCR_EL0.D is **RES0**.	0b0
[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** Number of counters implemented, 2 to 33, minus one. Must be configured to 0x06	0x06

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	31:0

This interface is accessible as follows:

When DoubleLockStatus() || !IsCorePowered() || OSLockStatus() || !AllowExternalPMUAccess()

ERROR

Otherwise

RO

B.3.38 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

PMCR_EL0 is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xE04

Access type

See bit descriptions

Reset value

```
0000 0000 0000 0000 0000 0xxx xxx0 x000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-60: pmu_pmcr_el0 bit assignments

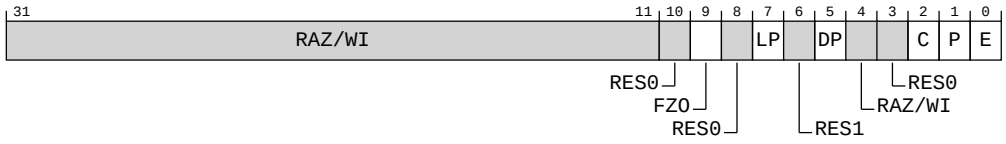


Table B-122: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[31:11]	RAZ/WI	Reserved	RAZ/WI
[10]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[9]	FZO	Freeze-on-overflow. Stop event counters on overflow. In the description of this field: • If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. • If EL2 is not implemented, PMN is PMCR_EL0.N. **0b0** Do not freeze on overflow. **0b1** Event counter PMU.PMEVCNTR<n>_EL0 does not count when AArch64-PMOVSCLR_EL0[(PMN-1):0] is nonzero and n is in the range of affected event counters.	x
[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 PMU.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 PMU.PMCCNTR_EL0 is not affected by this mechanism. **0b1** Cycle counting by PMU.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 PMU.PMCCNTR_EL0 is disabled at EL2. • If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and AArch64-MDCR_EL3.MPMX is 1, then cycle counting by PMU.PMCCNTR_EL0 is disabled at EL3. • If FEAT_PMUv3p7 is implemented and event counting is frozen by PMCR_EL0.FZO, then cycle counting by PMU.PMCCNTR_EL0 is disabled. • 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 PMU.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 PMU.PMCCNTR_EL0 to zero. Access to this field is: WO/**RAZ**	0b0

Bits	Name	Description	Reset
[1]	P	Event counter reset. The effects of writing to this bit are: **0b1** Reset all event counters, not including PMU.PMCCNTR_EL0, to zero. Access to this field is: WO/**RAZ**	0b0
[0]	E	Enable **0b1** PMU.PMCCNTR_EL0 and event counters PMU.PMEVCNTR<n>_EL0, where n is in the range of affected event counters, are enabled by PMU.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 DoubleLockStatus() || !IsCorePowered() || OSLockStatus() || !AllowExternalPMUAccess()
ERROR

When SoftwareLockStatus()
RO

Otherwise
RW

B.3.39 PMCEID0, Performance Monitors Common Event Identification register 0

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Note

This view of the register was previously called PMCEID0_EL0.

Configurations

PMCEID0 is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xE20

Access type

See bit descriptions

Reset value

0111 1111 1111 1111 0110 1111 0011 1111

Bit descriptions

Figure B-61: pmu_pmceid0 bit assignments

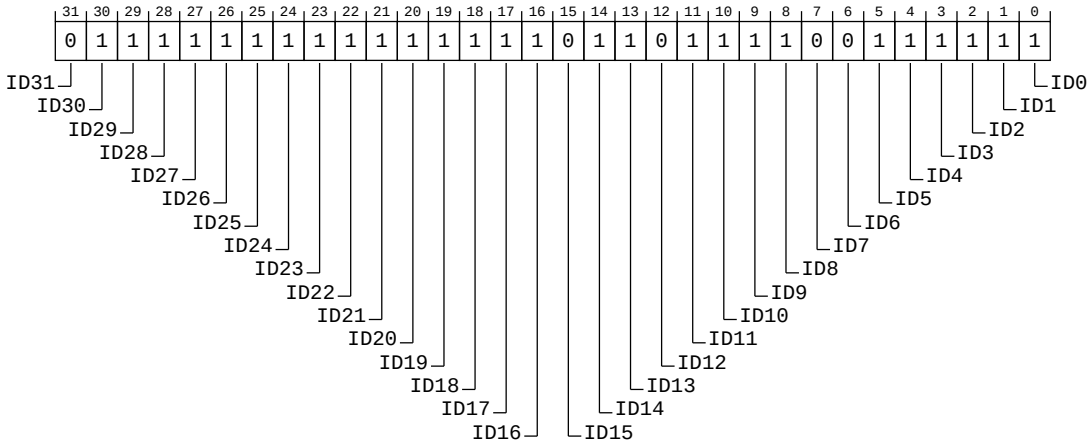


Table B-124: PMCEID0 bit descriptions

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

Bits	Name	Description	Reset
[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
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[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 **0b1** The Common event is implemented.	0b1
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **0b1** The Common event is implemented.	0b1
[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
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL **0b1** The Common event is implemented.	0b1

Bits	Name	Description	Reset
[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 DoubleLockStatus() || !IsCorePowered() || OSLockStatus() || !AllowExternalPMUAccess()
ERROR

Otherwise
RO

B.3.40 PMCEID1, Performance Monitors Common Event Identification register 1

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.


Note

This view of the register was previously called PMCEID1_EL0.

Configurations

PMCEID1 is in the Core power domain.

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-62: pmu_pmceid1 bit assignments

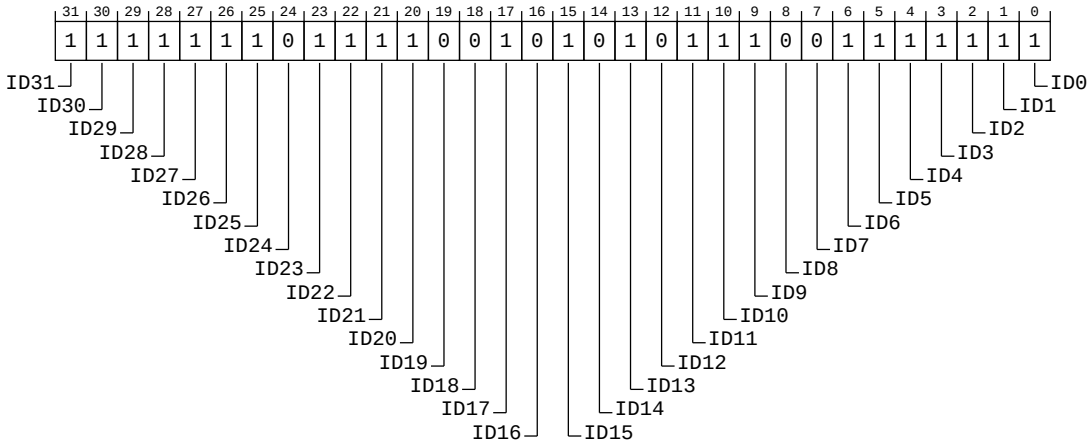


Table B-126: PMCEID1 bit descriptions

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

Bits	Name	Description	Reset
[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_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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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 DoubleLockStatus() || !IsCorePowered() || OSLockStatus() || !AllowExternalPMUAccess()

ERROR

Otherwise

RO

B.3.41 PMCEID2, Performance Monitors Common Event Identification register 2

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

PMCEID2 is in the Core power domain.

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-63: pmu_pmceid2 bit assignments

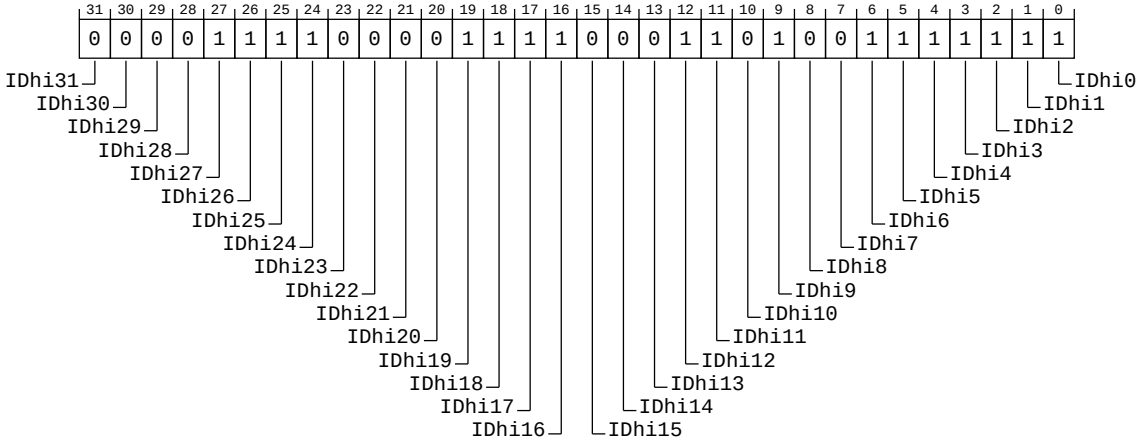


Table B-128: PMCEID2 bit descriptions

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

Bits	Name	Description	Reset
[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
[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

Bits	Name	Description	Reset
[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
[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 DoubleLockStatus() || !IsCorePowered() || OSLockStatus() || !AllowExternalPMUAccess()

ERROR

Otherwise

RO

B.3.42 PMCEID3, Performance Monitors Common Event Identification register 3

Defines which Common architectural events and Common microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

For more information about the Common events and the use of the PMCEIDn registers, see *The PMU event number space and common events* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

PMCEID3 is in the Core power domain.

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-64: pmu_pmceid3 bit assignments

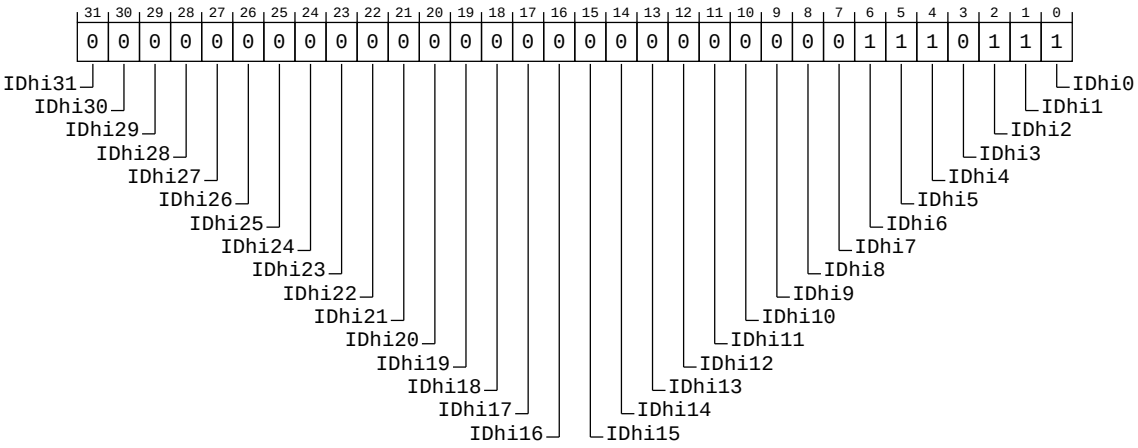


Table B-130: PMCEID3 bit descriptions

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 DoubleLockStatus() || !IsCorePowered() || OSLockStatus() || !AllowExternalPMUAccess()
ERROR

Otherwise
RO

B.3.43 PMSSCR, PMU Snapshot Capture Register

Provides a mechanism for software to initiate a sample.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
PMU

Register offset
0xE30

Access type
RESERVEDW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-65: ext_pmsscr bit assignments

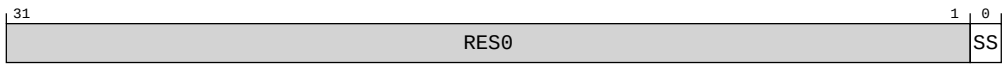


Table B-132: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	SS	Capture now. **0b0** Ignored. **0b1** Initiate a capture immediately.	x

Accessibility

Component	Offset	Range
PMU	0xE30	None

This interface is accessible as follows:

WO

B.3.44 PMMIR, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation.

Configurations

PMMIR is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xE40

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx 0000 0000 0000 0000 1010
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-66: pmu_pmmir bit assignments

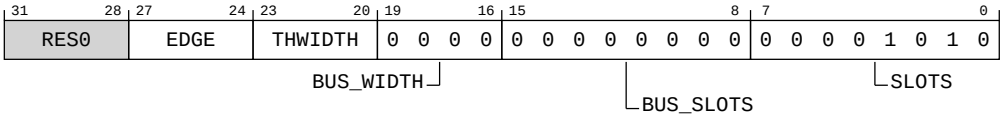


Table B-134: PMMIR bit descriptions

Bits	Name	Description	Reset
[31:28]	**RES0**	Reserved	**RES0**
[27:24]	EDGE	PMU event edge detection. Indicates implementation of the FEAT_PMUv3_EDGE feature. **0b0000** FEAT_PMUv3_EDGE is not implemented. **0b0001** FEAT_PMUv3_EDGE is implemented.	The reset values can be the following: 0b0000, 0b0001, respective to the value.

Bits	Name	Description	Reset
[23:20]	THWIDTH	PMU.PMEVTYPER<n>_EL0.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the PMU.PMEVTYPER<n>_EL0.TH field. **0b0000** FEAT_PMUv3_TH is not implemented. **0b0001** 1 bit. PMU.PMEVTYPER<n>_EL0.TH[11:1] are **RES0**. **0b0010** 2 bits. PMU.PMEVTYPER<n>_EL0.TH[11:2] are **RES0**. **0b0011** 3 bits. PMU.PMEVTYPER<n>_EL0.TH[11:3] are **RES0**. **0b0100** 4 bits. PMU.PMEVTYPER<n>_EL0.TH[11:4] are **RES0**. **0b0101** 5 bits. PMU.PMEVTYPER<n>_EL0.TH[11:5] are **RES0**. **0b0110** 6 bits. PMU.PMEVTYPER<n>_EL0.TH[11:6] are **RES0**. **0b0111** 7 bits. PMU.PMEVTYPER<n>_EL0.TH[11:7] are **RES0**. **0b1000** 8 bits. PMU.PMEVTYPER<n>_EL0.TH[11:8] are **RES0**. **0b1001** 9 bits. PMU.PMEVTYPER<n>_EL0.TH[11:9] are **RES0**. **0b1010** 10 bits. PMU.PMEVTYPER<n>_EL0.TH[11:10] are **RES0**. **0b1011** 11 bits. PMU.PMEVTYPER<n>_EL0.TH[11] is **RES0**. **0b1100** 12 bits.	The reset values can be the following: `0b0000`, `0b0001`, `0b0010`, `0b0011`, `0b0100`, `0b0101`, `0b0110`, `0b0111`, `0b1000`, `0b1001`, `0b1010`, `0b1011`, `0b1100`, respective to the value.
[19:16]	BUS_WIDTH	Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as Log_2(number of bytes), plus one. **0b0000** The information is not available.	`0b0000`

Bits	Name	Description	Reset
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. **0b00000000** The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is 0	0x00
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero. **0b00001010** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 10.	0x0A

Accessibility

If the Core power domain is off or in a low-power state, access on this interface returns an Error.

Component	Offset	Instance	Range
PMU	0xE40	PMMIR	31:0

This interface is accessible as follows:

When DoubleLockStatus() || !IsCorePowered() || OSLockStatus() || !AllowExternalPMUAccess()

ERROR

Otherwise

RO

B.3.45 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-67: pmu_pmdevarch bit assignments

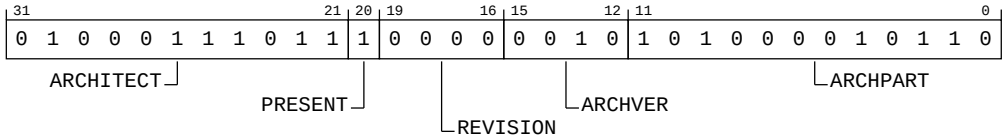


Table B-136: PMDEVARCH bit descriptions

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.	`0b0010`
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **`0b101000010110`** Armv8-A PE performance monitors.	`0xA16`

Accessibility

Component	Offset	Instance	Range
PMU	0xFBC	PMDEVARCH	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.3.46 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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-68: pmu_pmdevid bit assignments

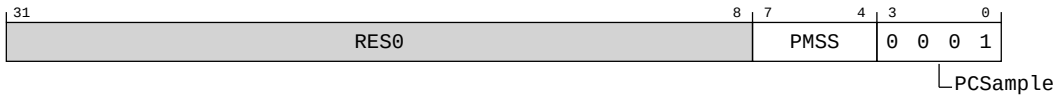


Table B-138: PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	PMSS	PMU Snapshot extension. Defined values are: **0b0000** PMU snapshot extension not implemented. **0b0001** PMU snapshot extension implemented.	The reset values can be the following: 0b0000, 0b0001, respective to the value.
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. **0b0001** PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	0b0001

Accessibility

Component	Offset	Instance	Range
PMU	0xFC8	PMDEVID	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.3.47 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFCC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0001 0110
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-69: pmu_pmdevtype bit assignments

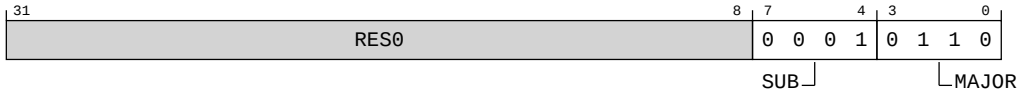


Table B-140: PMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SUB	Subtype. Indicates this is a component within a PE. **0b0001**	0b0001
[3:0]	MAJOR	Major type. Indicates this is a performance monitor component. **0b0110**	0b0110

Accessibility

Component	Offset	Instance	Range
PMU	0xFCC	PMDEVTYPE	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.3.48 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-70: pmu_pmpidr4 bit assignments

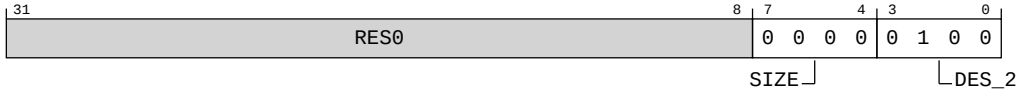


Table B-142: PMPIDR4 bit descriptions

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()
ERROR

Otherwise
RO

B.3.49 PMPIDR0, Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

If FEAT_DoPD is implemented, this register is in the Core power domain. If FEAT_DoPD is not implemented, this register is in the Debug power domain.

This register is required for CoreSight compliance.

Attributes

Width
32

Component
PMU

Register offset
0xFE0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-71: pmu_pmpidr0 bit assignments

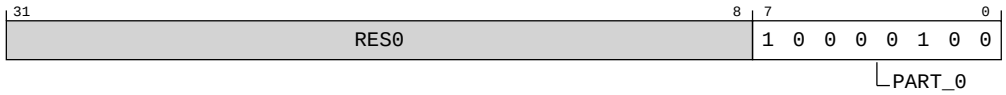


Table B-144: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b10000100** Least significant byte of the PMU unit part.	0x84

Accessibility

Component	Offset	Instance	Range
PMU	0xFE0	PMPIDR0	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.3.50 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-72: pmu_pmpidr1 bit assignments

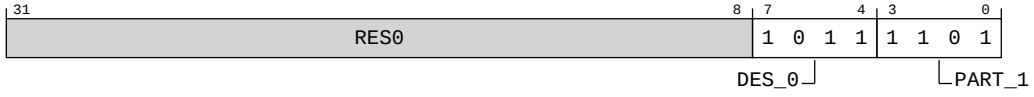


Table B-146: PMPIDR1 bit descriptions

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()

ERROR

Otherwise

RO

B.3.51 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 xxxx 0000 1011
|    |    |    |    |    |    |    |    |
31   27   23   19   15   11   7    3    0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-73: pmu_pmpidr2 bit assignments

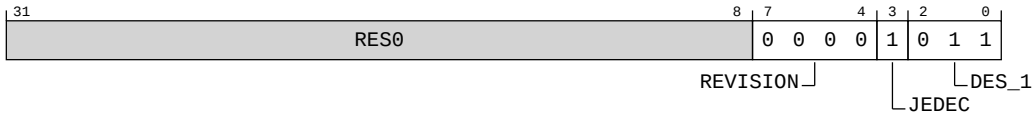


Table B-148: PMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0000** r0p1	0b0000
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited. 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()

ERROR

Otherwise

RO

B.3.52 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 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-74: pmu_pmpidr3 bit assignments

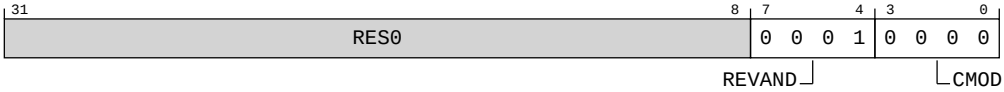


Table B-150: PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVAND	Part minor revision. Parts using PMU.PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0001**	0b0001
[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()

ERROR

Otherwise

RO

B.3.53 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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-75: pmu_pmcidr0 bit assignments

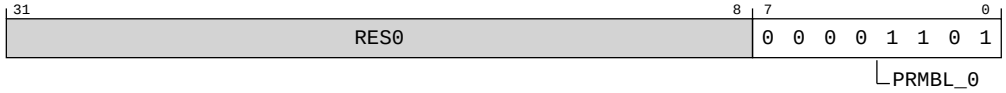


Table B-152: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_0	Preamble. **0b00001101**	0x0D

Accessibility

Component	Offset	Instance	Range
PMU	0xFF0	PMCIDR0	None

This interface is accessible as follows:

When !IsCorePowered()

ERROR

Otherwise

RO

B.3.54 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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-76: pmu_pmcidr1 bit assignments

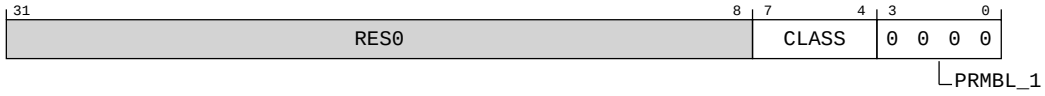


Table B-154: PMCIDR1 bit descriptions

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()

ERROR

Otherwise

RO

B.3.55 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-77: pmu_pmcidr2 bit assignments

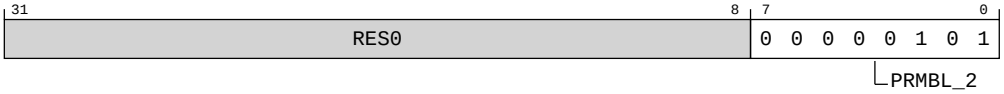


Table B-156: PMCIDR2 bit descriptions

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()

ERROR

Otherwise

RO

B.3.56 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-78: pmu_pmcidr3 bit assignments

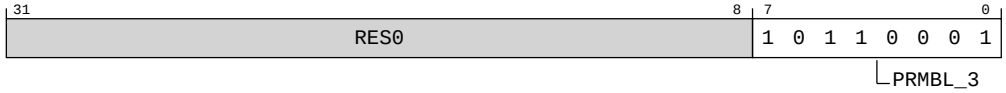


Table B-158: PMCIDR3 bit descriptions

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()

ERROR

Otherwise
RO

B.4 External Debug registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table B-160: Debug registers summary

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

Offset	Name	Reset	Width	Description
0x458	DBGBCR5_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x800	DBGWVR0_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x808	DBGWCR0_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x810	DBGWVR1_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x818	DBGWCR1_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x820	DBGWVR2_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x828	DBGWCR2_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0x830	DBGWVR3_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x838	DBGWCR3_EL1	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	See individual bit resets.	32-bit	Main ID Register
0xD20	EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
0xD28	EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
0xD48	EDDFR1 [31:0]	See individual bit resets.	32-bit	External Debug Feature Register 1
0xD4C	EDDFR1 [63:32]	See individual bit resets.	32-bit	External Debug Feature Register 1
0xD60	EDAA32PFR	See individual bit resets.	64-bit	External Debug Auxiliary Processor Feature Register
0xF00	EDITCTRL	See individual bit resets.	32-bit	External Debug Integration mode Control register
0xFA0	DBGCLAIMSET_EL1	See individual bit resets.	32-bit	Debug CLAIM Tag Set register
0xFA4	DBGCLAIMCLR_EL1	See individual bit resets.	32-bit	Debug CLAIM Tag Clear register
0xFA8	EDDEVAFF0	See individual bit resets.	32-bit	External Debug Device Affinity register 0
0xFAC	EDDEVAFF1	See individual bit resets.	32-bit	External Debug Device Affinity register 1
0xFB0	EDLAR	See individual bit resets.	32-bit	External Debug Lock Access Register
0xFB4	EDLSR	See individual bit resets.	32-bit	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	See individual bit resets.	32-bit	Debug Authentication Status register
0xFBC	EDDEVARCH	See individual bit resets.	32-bit	External Debug Device Architecture register
0xFC0	EDDEVID2	See individual bit resets.	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	See individual bit resets.	32-bit	External Debug Device ID register 1
0xFC8	EDDEVID	See individual bit resets.	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	See individual bit resets.	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	See individual bit resets.	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	See individual bit resets.	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	See individual bit resets.	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	See individual bit resets.	32-bit	External Debug Component Identification Register 3

B.4.1 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in ext-EDSCR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0x090

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-79: ext_edrcr bit assignments

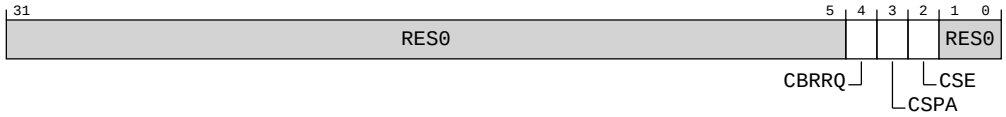


Table B-161: EDRCR bit descriptions

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.4.2 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

Configurations

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR_EL1.

Attributes

Width
32

Component
Debug

Register offset
0x310

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-80: ext_edprcr bit assignments

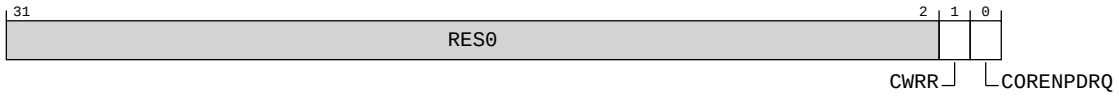


Table B-163: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:2]	**RES0**	Reserved	**RES0**
[1]	CWRR	**When IsFeatureImplemented(FEAT_RME)** The PE ignores all writes to this bit. **Otherwise** Warm reset request. The extent of the reset is **IMPLEMENTATION DEFINED**, but must be one of: • The request is ignored. - Only this PE is Warm reset. - This PE and other components of the system, possibly including other PEs, are Warm reset. Arm deprecates use of this bit, and recommends that implementations ignore the request. **0b0** No action. **0b1** Request Warm reset.	x

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 [6]

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.4.3 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.

[6] On a Cold reset, if the powerup request is implemented and the powerup request has been asserted, this field is an **IMPLEMENTATION DEFINED** choice of 0 or 1. If the powerup request is not asserted, this field is set to 0.

- When 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-81: ext_dbgbvr0_el1 bit assignments

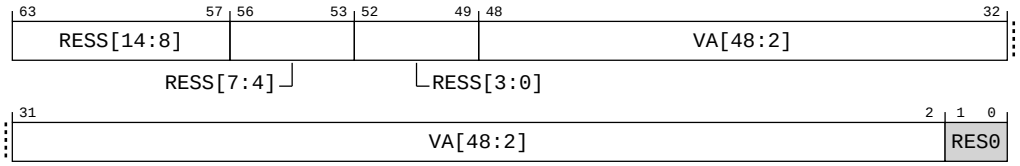


Table B-165: DBGBVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-82: ext_dbgbvr0_el1 bit assignments

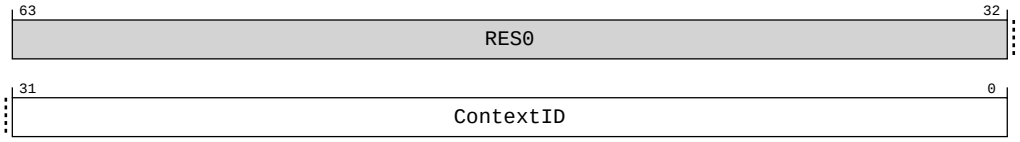


Table B-166: DBGBVR0_EL1 bit descriptions

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), 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-83: ext_dbgbvr0_el1 bit assignments

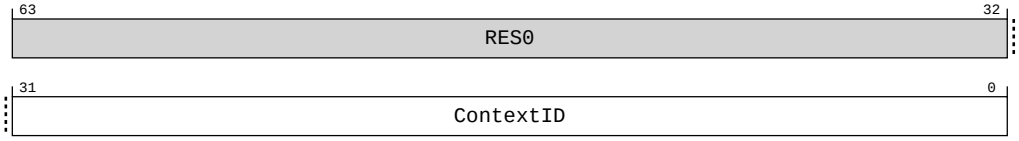


Table B-167: DBGBVR0_EL1 bit descriptions

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-84: ext_dbgbvr0_el1 bit assignments

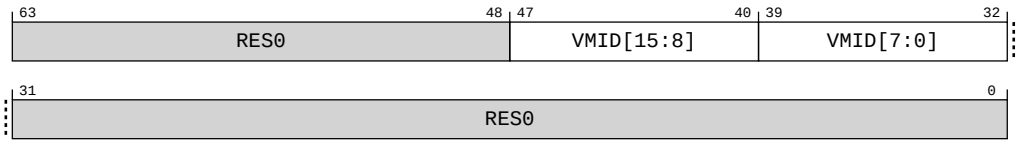


Table B-168: DBGBVR0_EL1 bit descriptions

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}

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 ext-DBGBCR0_EL1.BT == '101'

Figure B-85: ext_dbgbvr0_el1 bit assignments

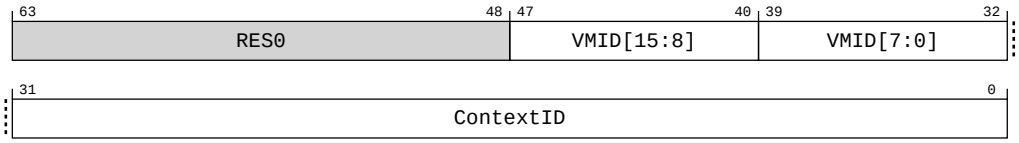


Table B-169: DBGBVR0_EL1 bit descriptions

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-86: ext_dbgbvr0_el1 bit assignments

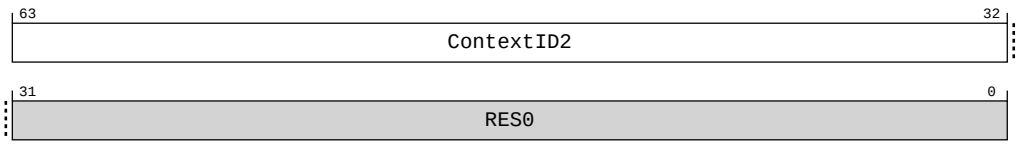


Table B-170: DBGBVR0_EL1 bit descriptions

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 ext-DBGBCR0_EL1.BT == '111'

Figure B-87: ext_dbgbvr0_el1 bit assignments

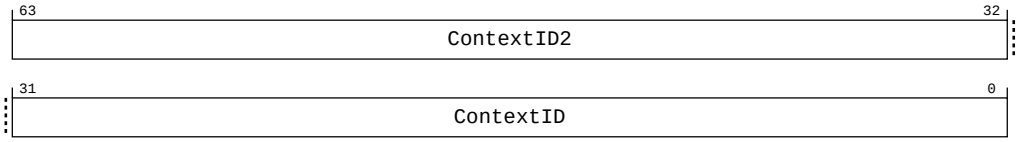


Table B-171: DBGBVR0_EL1 bit descriptions

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.4.4 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

64

Component

Debug

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-88: ext_dbgbcr0_el1 bit assignments

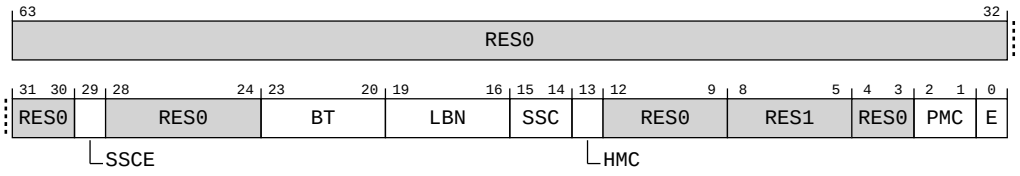


Table B-173: DBGBCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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

Bits	Name	Description	Reset
[0]	E	Enable breakpoint n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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.4.5 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 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-89: ext_dbgbvr1_el1 bit assignments

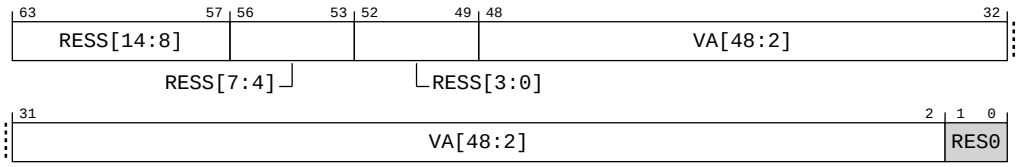


Table B-175: DBGBVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-90: ext_dbgbvr1_el1 bit assignments

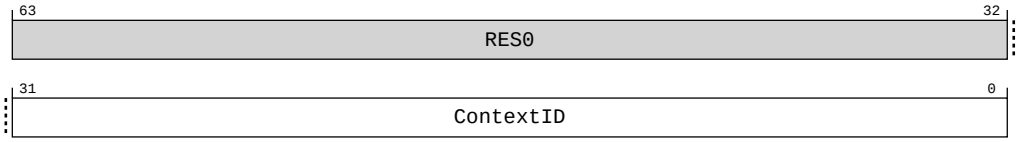


Table B-176: DBGBVR1_EL1 bit descriptions

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-91: ext_dbgbvr1_el1 bit assignments

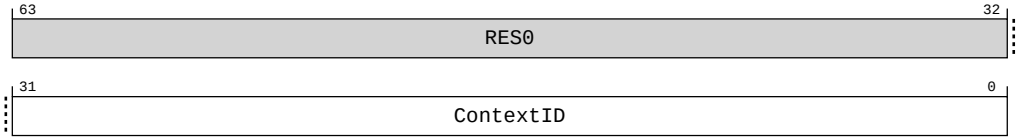


Table B-177: DBGBVR1_EL1 bit descriptions

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-92: ext_dbgbvr1_el1 bit assignments

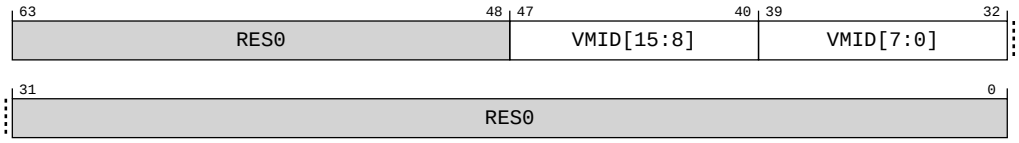


Table B-178: DBGBVR1_EL1 bit descriptions

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-93: ext_dbgbvr1_el1 bit assignments

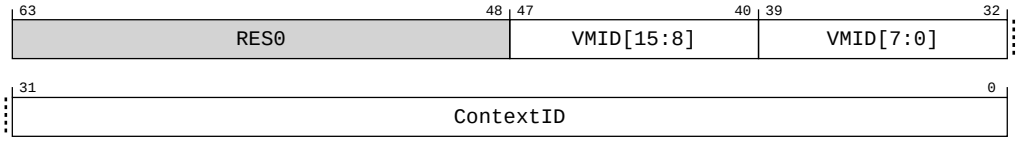


Table B-179: DBGBVR1_EL1 bit descriptions

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-94: ext_dbgbvr1_el1 bit assignments

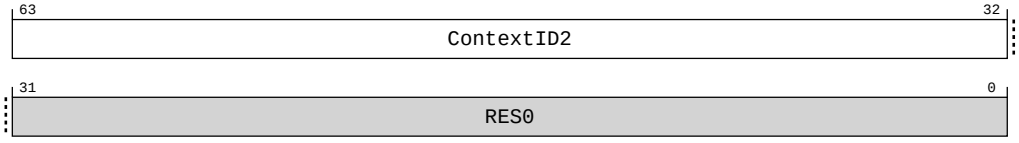


Table B-180: DBGBVR1_EL1 bit descriptions

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-95: ext_dbgbvr1_el1 bit assignments

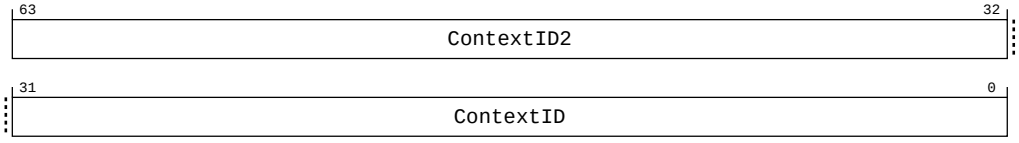


Table B-181: DBGBVR1_EL1 bit descriptions

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.4.6 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

64

Component

Debug

Register offset

0x418

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-96: ext_dbgbcr1_el1 bit assignments

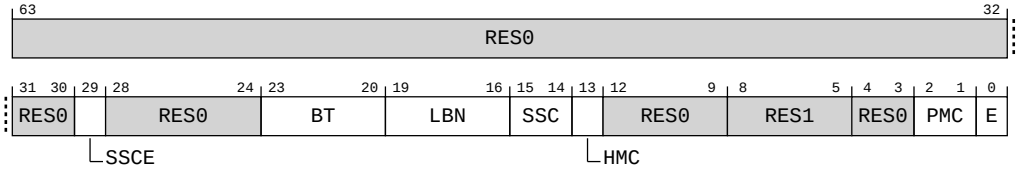


Table B-183: DBGBCR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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.4.7 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-97: ext_dbgbvr2_el1 bit assignments

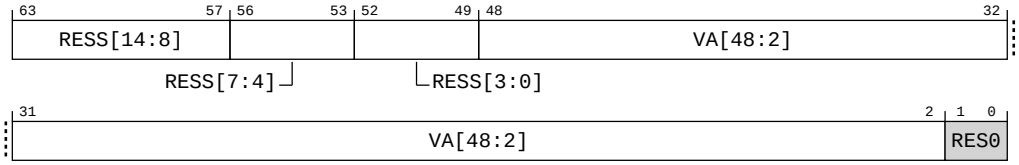


Table B-185: DBGBVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}

Bits	Name	Description	Reset
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-98: ext_dbgbvr2_el1 bit assignments

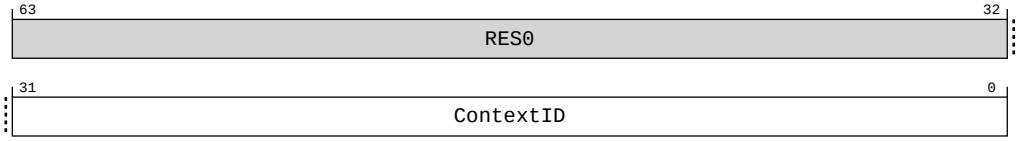


Table B-186: DBGBVR2_EL1 bit descriptions

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-99: ext_dbgbvr2_el1 bit assignments

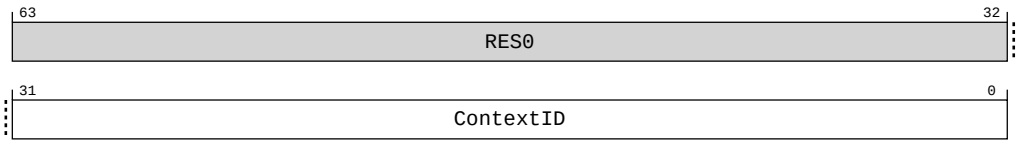


Table B-187: DBGBVR2_EL1 bit descriptions

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-100: ext_dbgbvr2_el1 bit assignments

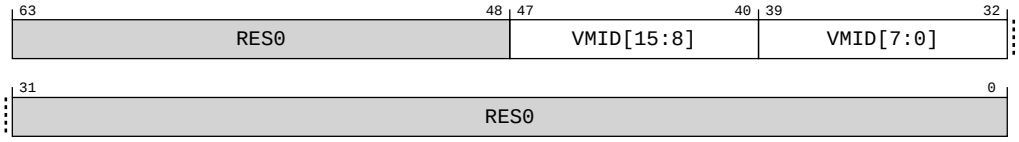


Table B-188: DBGBVR2_EL1 bit descriptions

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-101: ext_dbgbvr2_el1 bit assignments

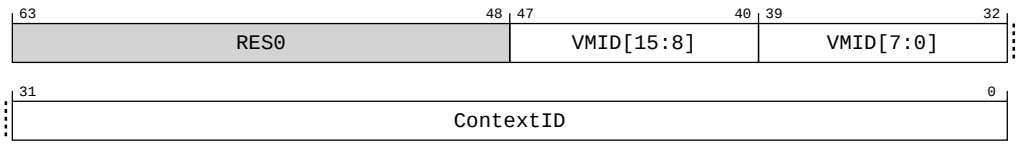


Table B-189: DBGBVR2_EL1 bit descriptions

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-102: ext_dbgbvr2_el1 bit assignments

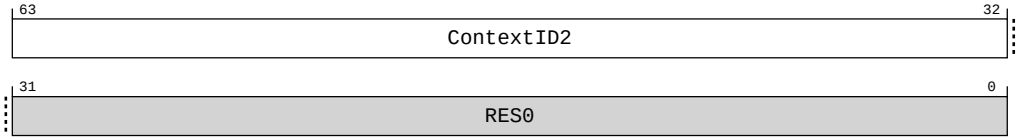


Table B-190: DBGBVR2_EL1 bit descriptions

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-103: ext_dbgbvr2_el1 bit assignments

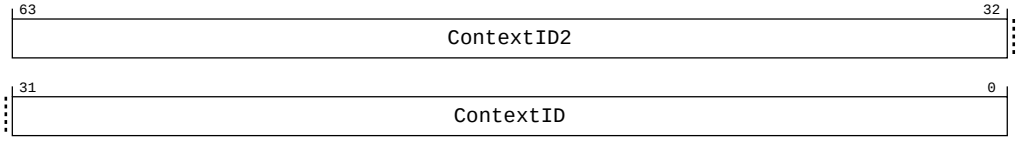


Table B-191: DBGBVR2_EL1 bit descriptions

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.4.8 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

64

Component

Debug

Register offset

0x428

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-104: ext_dbgbcr2_el1 bit assignments

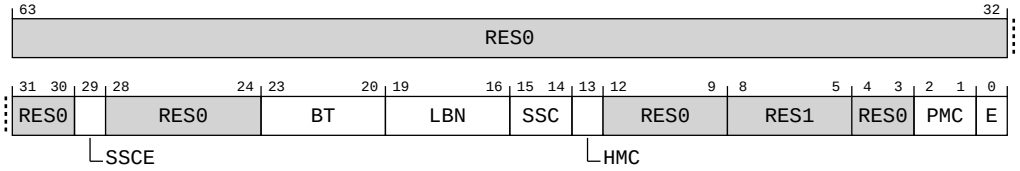


Table B-193: DBGBCR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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**

Bits	Name	Description	Reset
[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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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.4.9 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-105: ext_dbgbvr3_el1 bit assignments

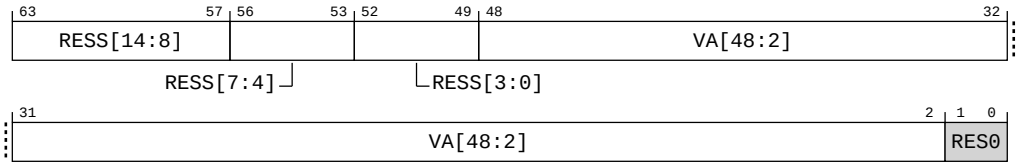


Table B-195: DBGBVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-106: ext_dbgbvr3_el1 bit assignments

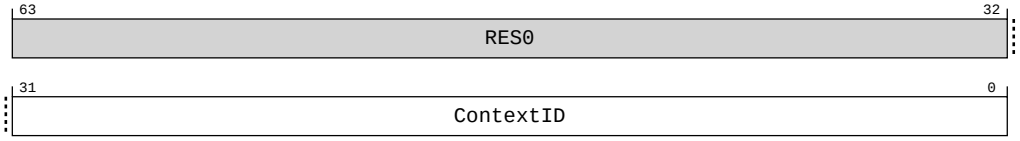


Table B-196: DBGBVR3_EL1 bit descriptions

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), 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-107: ext_dbgbvr3_el1 bit assignments

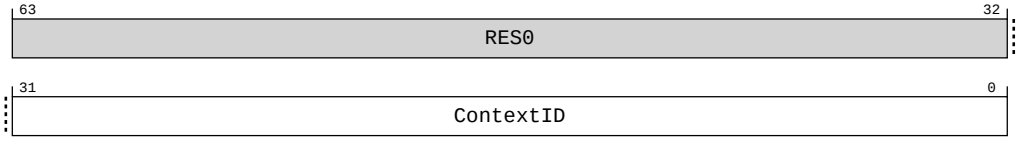


Table B-197: DBGBVR3_EL1 bit descriptions

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-108: ext_dbgbvr3_el1 bit assignments

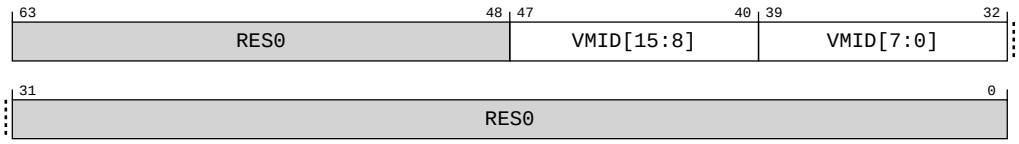


Table B-198: DBGBVR3_EL1 bit descriptions

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 ext-DBGBCR3_EL1.BT == '101'

Figure B-109: ext_dbgbvr3_el1 bit assignments

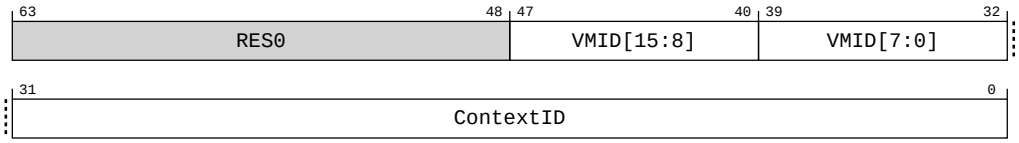


Table B-199: DBGBVR3_EL1 bit descriptions

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-110: ext_dbgbvr3_el1 bit assignments

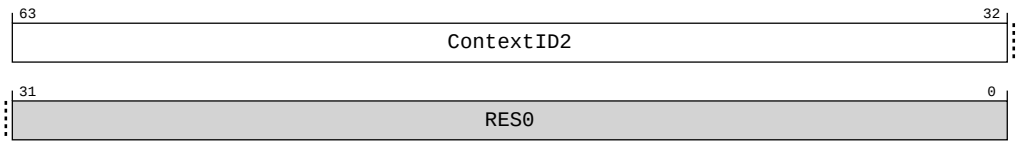


Table B-200: DBGBVR3_EL1 bit descriptions

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 ext-DBGBCR3_EL1.BT == '111'

Figure B-111: ext_dbgbvr3_el1 bit assignments

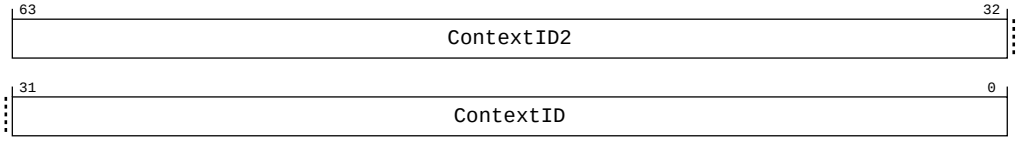


Table B-201: DBGBVR3_EL1 bit descriptions

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.4.10 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

64

Component

Debug

Register offset

0x438

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-112: ext_dbgbcr3_el1 bit assignments

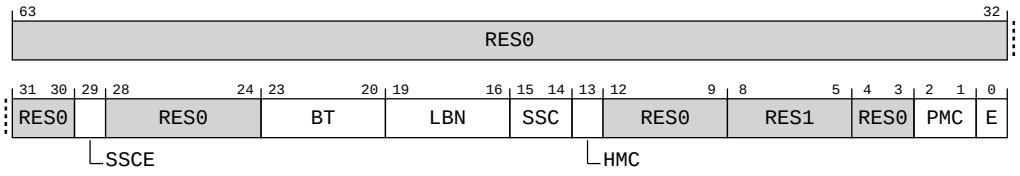


Table B-203: DBGBCR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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

Bits	Name	Description	Reset
[0]	E	Enable breakpoint n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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.4.11 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-113: ext_dbgbvr4_el1 bit assignments

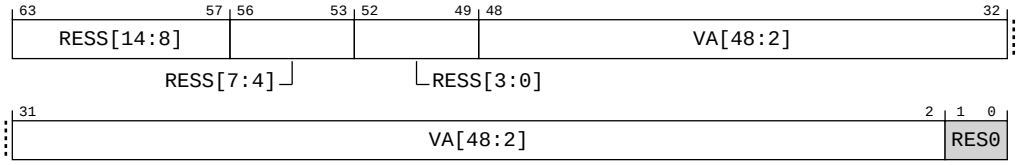


Table B-205: DBGBVR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-114: ext_dbgbvr4_el1 bit assignments

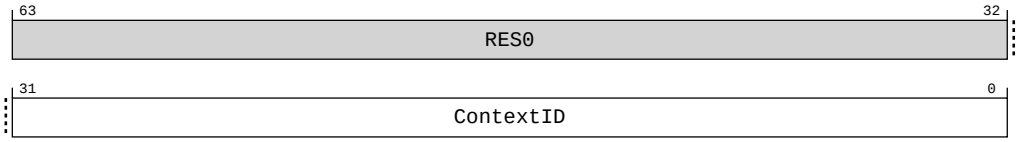


Table B-206: DBGBVR4_EL1 bit descriptions

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-115: ext_dbgbvr4_el1 bit assignments

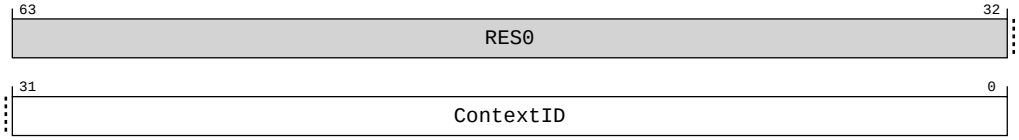


Table B-207: DBGBVR4_EL1 bit descriptions

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-116: ext_dbgbvr4_el1 bit assignments

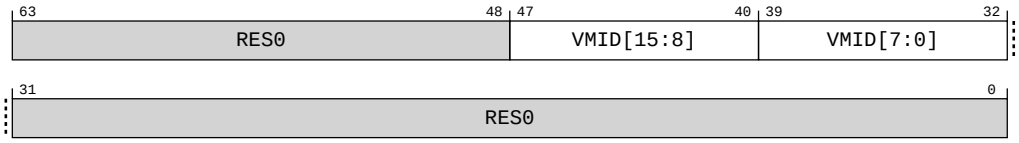


Table B-208: DBGBVR4_EL1 bit descriptions

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-117: ext_dbgbvr4_el1 bit assignments

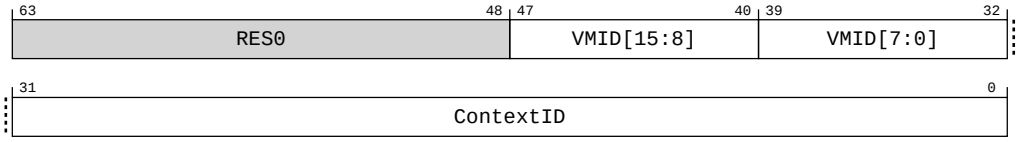


Table B-209: DBGBVR4_EL1 bit descriptions

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-118: ext_dbgbvr4_el1 bit assignments

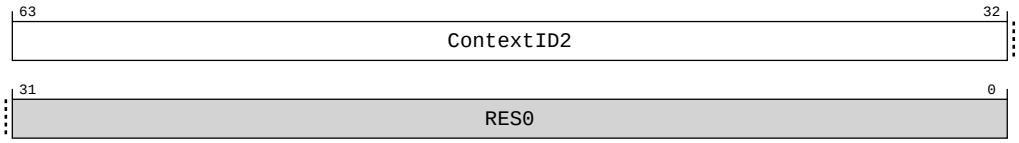


Table B-210: DBGBVR4_EL1 bit descriptions

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-119: ext_dbgbvr4_el1 bit assignments

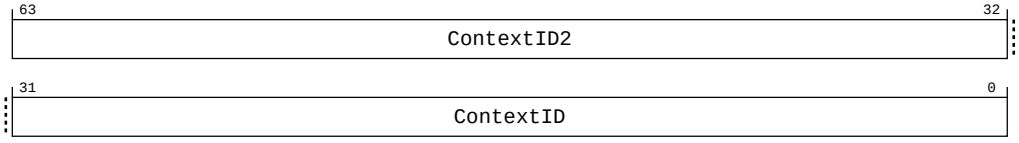


Table B-211: DBGBVR4_EL1 bit descriptions

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.4.12 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

64

Component

Debug

Register offset

0x448

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-120: ext_dbgbcr4_el1 bit assignments

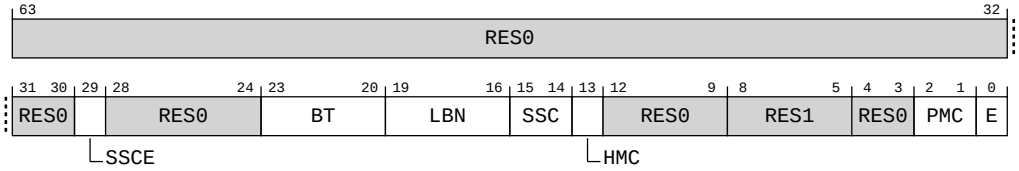


Table B-213: DBGBCR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx

Bits	Name	Description	Reset
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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.4.13 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-121: ext_dbgbvr5_el1 bit assignments

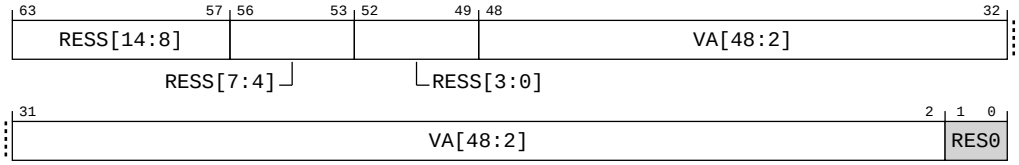


Table B-215: DBGBVR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}

Bits	Name	Description	Reset
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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-122: ext_dbgbvr5_el1 bit assignments

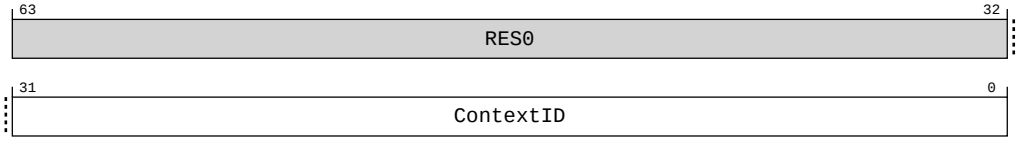


Table B-216: DBGBVR5_EL1 bit descriptions

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-123: ext_dbgbvr5_el1 bit assignments

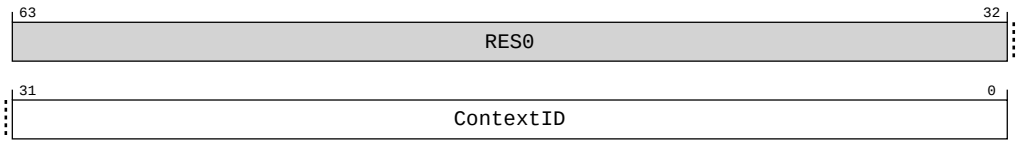


Table B-217: DBGBVR5_EL1 bit descriptions

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-124: ext_dbgbvr5_el1 bit assignments

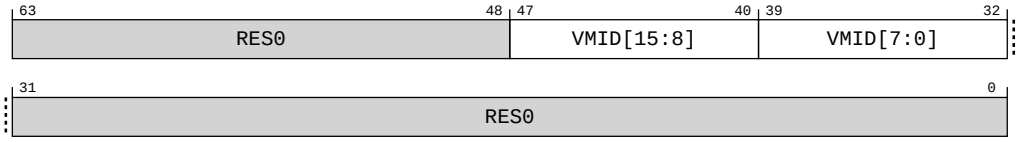


Table B-218: DBGBVR5_EL1 bit descriptions

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-125: ext_dbgbvr5_el1 bit assignments

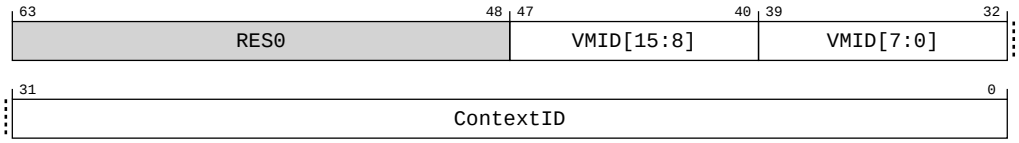


Table B-219: DBGBVR5_EL1 bit descriptions

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-126: ext_dbgbvr5_el1 bit assignments

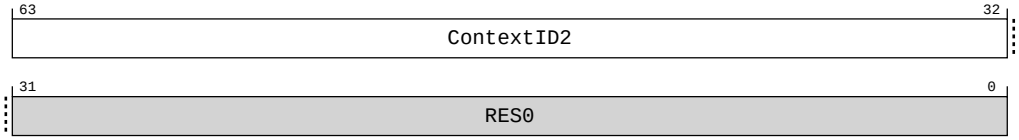


Table B-220: DBGBVR5_EL1 bit descriptions

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-127: ext_dbgbvr5_el1 bit assignments

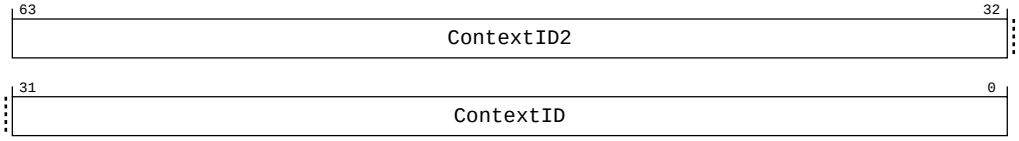


Table B-221: DBGBVR5_EL1 bit descriptions

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.4.14 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

64

Component

Debug

Register offset

0x458

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

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-128: ext_dbgbcr5_el1 bit assignments

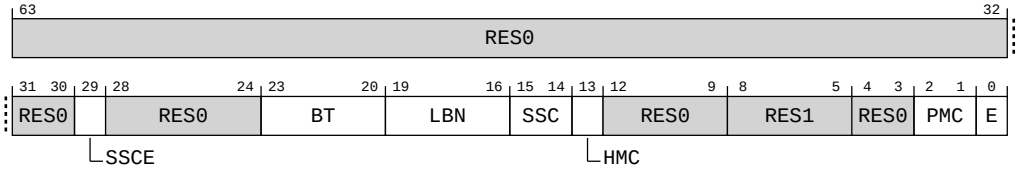


Table B-223: DBGBCR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the breakpoint can be generated are:HMC, PMC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	**RES0**	Reserved	**RES0**
[23:20]	BT	Breakpoint Type. With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type. **0b0000** Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction. **0b0001** Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.	xxxx
[19:16]	LBN	Linked Breakpoint Number. For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other breakpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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**

Bits	Name	Description	Reset
[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 n. **0b0** Breakpoint n disabled. **0b1** Breakpoint n 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.4.15 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-129: ext_dbgwvr0_el1 bit assignments

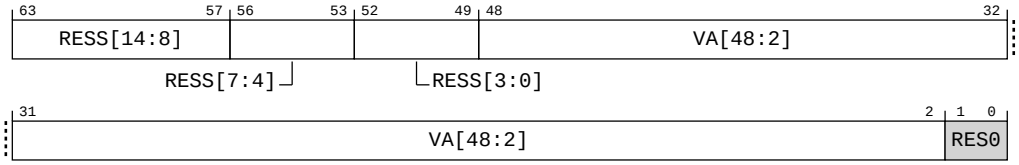


Table B-225: DBGWVR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx

Bits	Name	Description	Reset
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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.4.16 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

64

Component

Debug

Register offset

0x808

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-130: ext_dbgwcr0_el1 bit assignments

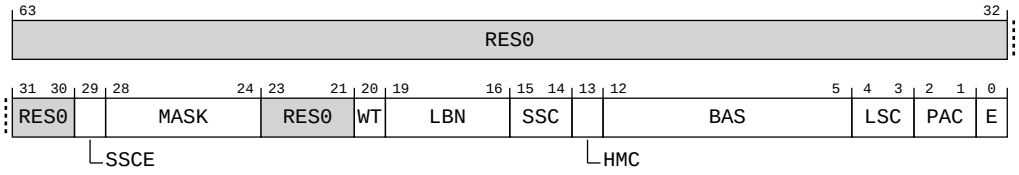


Table B-227: DBGWCR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	5{x}
[23:21]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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
[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-228: BAS description on page 1237 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-229: BAS description table 3 on page 1237 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 nonzero 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.	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

Bits	Name	Description	Reset
[0]	E	Enable watchpoint n. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table B-228: BAS description

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-229: BAS description table 3

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.4.17 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-131: ext_dbgwvr1_el1 bit assignments

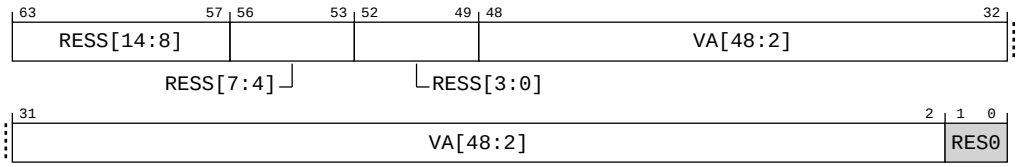


Table B-231: DBGWVR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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.4.18 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

64

Component

Debug

Register offset

0x818

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-132: ext_dbgwcr1_el1 bit assignments

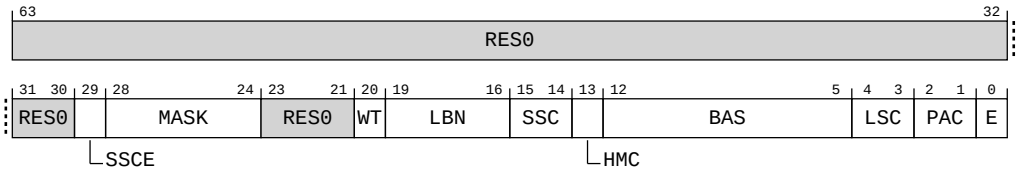


Table B-233: DBGWCR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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
[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-234: BAS description on page 1242 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-235: BAS description table 3 on page 1242 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 nonzero 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}

Bits	Name	Description	Reset
[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.	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. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table B-234: BAS description

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-235: BAS description table 3

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.4.19 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-133: ext_dbgwvr2_el1 bit assignments

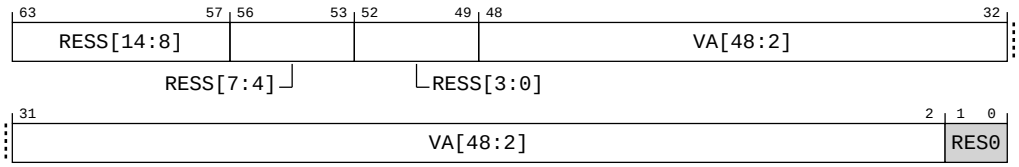


Table B-237: DBGWVR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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.4.20 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

64

Component

Debug

Register offset

0x828

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

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-134: ext_dbgwcr2_el1 bit assignments

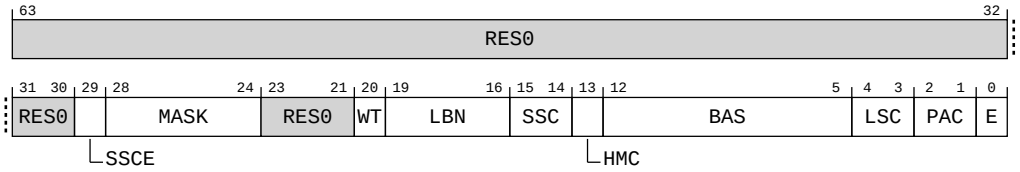


Table B-239: DBGWCR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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-240: BAS description on page 1247 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-241: BAS description table 3 on page 1247 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 nonzero 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.	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. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table B-240: BAS description

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-241: BAS description table 3

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.4.21 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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-135: ext_dbgwvr3_el1 bit assignments

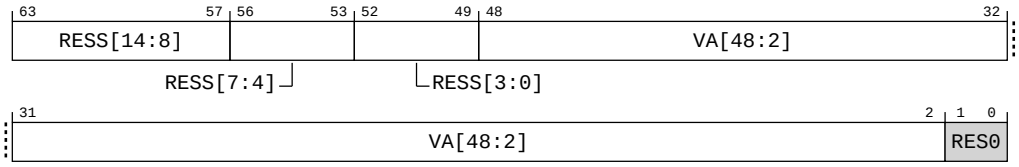


Table B-243: DBGWVR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RESS[14:8]	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.	7{x}
[56:53]	RESS[7:4]	Extension to RESS[14:8]. For more information, see RESS[14:8].	xxxx
[52:49]	RESS[3:0]	Extension to RESS[14:8]. For more information, see RESS[14:8].	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.4.22 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

64

Component

Debug

Register offset

0x838

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When the E field is zero, all the other fields in the register are ignored.

Figure B-136: ext_dbgwcr3_el1 bit assignments

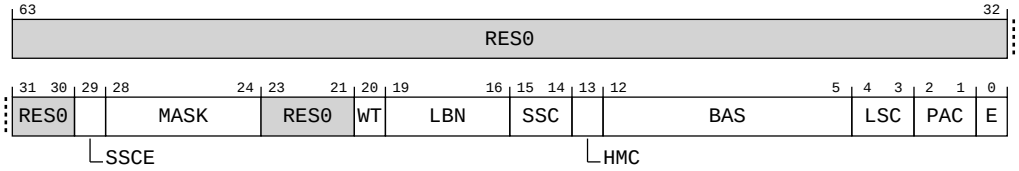


Table B-245: DBGWCR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:30]	**RES0**	Reserved	**RES0**
[29]	SSCE	**When IsFeatureImplemented(FEAT_RME)** Security State Control Extended. The fields that indicate when the watchpoint can be generated are:HMC, PAC, SSC, and SSCE.These fields must be considered in combination, and the values that are permitted for these fields are constrained. **Otherwise** RES0	x
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. **0b00000** No mask.	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, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to. For all other watchpoint types, this field is ignored and reads of the register return an **UNKNOWN** value.	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

Bits	Name	Description	Reset
[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
[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-246: BAS description on page 1252 In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-247: BAS description table 3 on page 1252 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 nonzero 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.	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. **0b0** Watchpoint n disabled. **0b1** Watchpoint n enabled.	x

Table B-246: BAS description

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-247: BAS description table 3

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

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
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.4.23 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 0000 1111 1101 1000 0100 0001

Bit descriptions

Figure B-137: ext_midr_el1 bit assignments

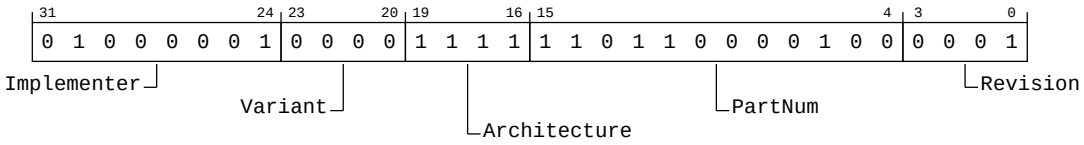


Table B-249: MIDR_EL1 bit descriptions

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. **0b0000** r0p1	0b0000
[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. **0b110110000100** Neoverse V3	0xD84
[3:0]	Revision	Revision number for the device. **0b0001** r0p1	0b0001

Accessibility

Component	Offset	Instance	Range
Debug	0xD00	MIDR_EL1	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.4.24 EDPFR, External Debug Processor Feature Register

Provides information about implemented PE features.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD20

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx 0001 xxxx 0001 0001 xxxx xxxx 0001 0001 0001 0001 0001 0001
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-138: ext_edpfr bit assignments

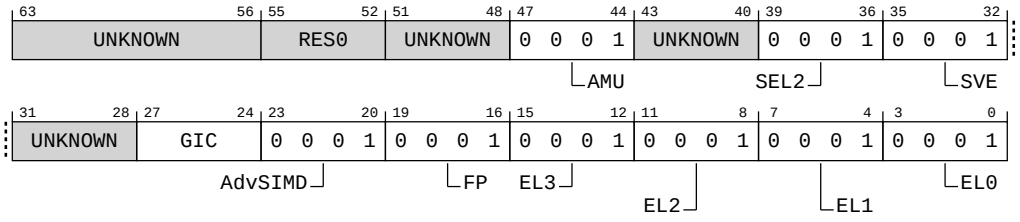


Table B-251: EDPFR bit descriptions

Bits	Name	Description	Reset
[63:56]	UNKNOWN	Reserved	UNKNOWN
[55:52]	RES0	Reserved	RES0
[51:48]	UNKNOWN	Reserved	UNKNOWN
[47:44]	AMU	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. 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. 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	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.4.25 EDDFR, External Debug Feature Register

Provides top level information about the debug system.


Note

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 offset

0xD28

Access type

See bit descriptions

Reset value

```
xxxx 0000 xxxx xxxx xxxx 0001 xxxx xxxx 0001 0000 0011 0000 0101 0111 0001 xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-139: ext_eddfr bit assignments

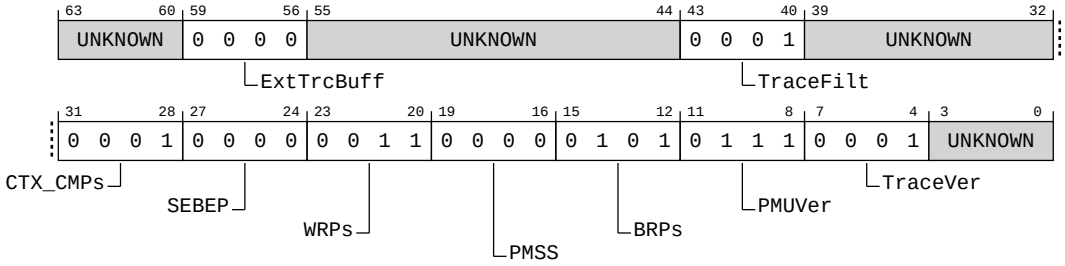


Table B-253: EDDFR bit descriptions

Bits	Name	Description	Reset
[63:60]	UNKNOWN	Reserved	UNKNOWN
[59:56]	ExtTrcBuff	Trace Buffer External Mode Extension. Defined values are: **0b0000** Trace Buffer Extension not implemented or Trace Buffer External Mode not implemented.	0b0000
[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. **0b0001** Two context-aware breakpoints are included	0b0001
[27:24]	SEBEP	This field either has the same value as AArch64-ID_AA64DFR0_EL1.SEBEP or reads as zero. **0b0000** Synchronous-exception-based event profiling not implemented.	0b0000
[23:20]	WRPs	Number of watchpoints, minus 1. **0b0011** Four Watchpoints	0b0011
[19:16]	PMSS	This field either has the same value as AArch64-ID_AA64DFR0_EL1.PMSS or reads as zero. **0b0000** PMU snapshot extension not implemented.	0b0000
[15:12]	BRPs	Number of breakpoints, minus 1. **0b0101** Six Breakpoints	0b0101

Bits	Name	Description	Reset
[11:8]	PMUVer	Performance Monitors Extension version. This field does not follow the standard ID scheme, but uses the alternative ID scheme described in *Alternative ID scheme used for the Performance Monitors Extension version* in the *Arm® Architecture Reference Manual for A-profile architecture* Defined values are: **0b0111** PMUv3 for Armv8.7. As 0b0110, and adds support for: • The PMU.PMCR_EL0.FZO and, if EL2 is implemented, AArch64-MDCR_EL2.HPMFZO controls. • If EL3 is implemented, the AArch64-MDCR_EL3.{MPMX,MCCD} controls.	0b0111
[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	None

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()
RO

Otherwise
ImplementationDefined

B.4.26 EDDFR1, External Debug Feature Register 1

Provides top level information about the debug system in AArch64.

Configurations

This register is available in all configurations.

Attributes

Width
64

Component
Debug

Register offsets (2)
0xD48,0xD4C

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-140: ext_eddfr1 bit assignments

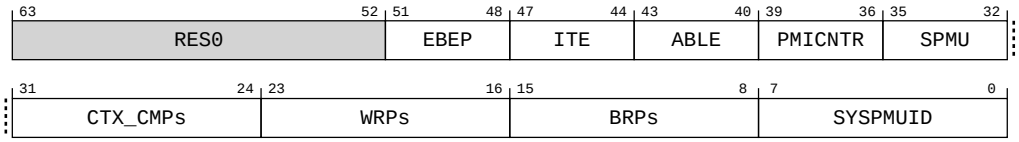


Table B-255: EDDFR1 bit descriptions

Bits	Name	Description	Reset
[63:52]	RES0	Reserved	RES0
[51:48]	EBEP	This field either has the same value as AArch64-ID_AA64DFR1_EL1.EBEP or reads as zero.	xxxx
[47:44]	ITE	This field either has the same value as AArch64-ID_AA64DFR1_EL1.ITE or reads as zero.	xxxx
[43:40]	ABLE	Address Breakpoint Linking Extension. Defined values are: **0b0000** Address Breakpoint Linking Extension not implemented. **0b0001** Address Breakpoint Linking Extension implemented.	xxxx
[39:36]	PMICNTR	This field either has the same value as AArch64-ID_AA64DFR1_EL1.PMICNTR or reads as zero.	xxxx
[35:32]	SPMU	This field either has the same value as AArch64-ID_AA64DFR1_EL1.SPMU or reads as zero.	xxxx
[31:24]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. The value 0x00 means that the number of breakpoints that are context-aware is described by ext-EDDFR.CTX_CMPs. Otherwise, the value of this field is the number of breakpoints that are context-aware, minus 1. Defined values are: **0b00000000** ext-EDDFR.CTX_CMPs is the number of breakpoints that are context-aware, minus 1.	8{x}
[23:16]	WRPs	Number of watchpoints, minus 1. The value 0x00 means that the number of watchpoints is described by ext-EDDFR.WRPs. Otherwise, the value of this field is the number of watchpoints, minus 1. Defined values are: **0b00000000** ext-EDDFR.WRPs is the number of watchpoints, minus 1.	8{x}

Bits	Name	Description	Reset
[15:8]	BRPs	Number of breakpoints, minus 1. The value `0x00` means that the number of breakpoints is described by ext-EDDFR.BRPs. Otherwise, the value of this field is the number of breakpoints, minus 1. Defined values are: **`0b00000000`** ext-EDDFR.BRPs is the number of breakpoints, minus 1.	`8{x}`
[7:0]	SYSPMUID	This field either has the same value as AArch64-ID_AA64DFR1_EL1.SYSPMUID or reads as zero.	`8{x}`

Accessibility

Component	Offset	Instance	Range
Debug	0xD48	EDDFR1	31:0

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

Component	Offset	Instance	Range
Debug	0xD4C	EDDFR1	63:32

This interface is accessible as follows:

When IsCorePowered() && !DoubleLockStatus()

RO

Otherwise

ImplementationDefined

B.4.27 EDDEVARCH, External Debug Device Architecture register

Identifies the programmers' model architecture of the external debug component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFBC

Access type

See bit descriptions

Reset value

0100 0111 0111 0000 1001 1010 0001 0101

Bit descriptions

Figure B-141: ext_eddevarch bit assignments

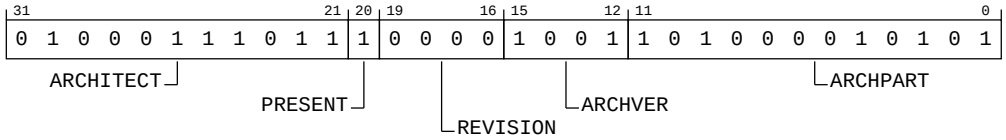


Table B-258: EDDEVARCH bit descriptions

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.4.28 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFC0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-142: ext_eddevid2 bit assignments

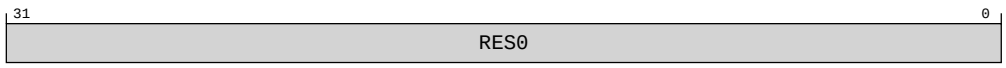


Table B-260: EDDEVID2 bit descriptions

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.4.29 EDDEVID1, External Debug Device ID register 1

Provides extra information for external debuggers about features of the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFC4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-143: ext_eddevid1 bit assignments

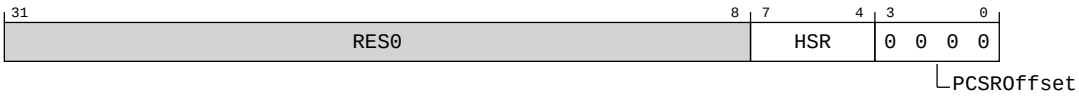


Table B-262: EDDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	HSR	Indicates support for the External Debug Halt Status Register, ext-EDHSR. Defined values are: **0b0000** ext-EDHSR not implemented, and the PE follows behaviors consistent with all of the ext-EDHSR fields having a zero value. **0b0001** ext-EDHSR implemented. **0b0010** As 0b0001, but extends ext-EDHSR to include the VNCR, CM, and WnR fields.	The reset values can be the following: `0b0000`, `0b0001`, `0b0010`, respective to the value.
[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.4.30 EDDEVID, External Debug Device ID register 0

Provides extra information for external debuggers about features of the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFC8

Access type

See bit descriptions

Reset value

```
xxxx 0000 xxxx xxxx xxxx xxxx 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-144: ext_eddevid bit assignments

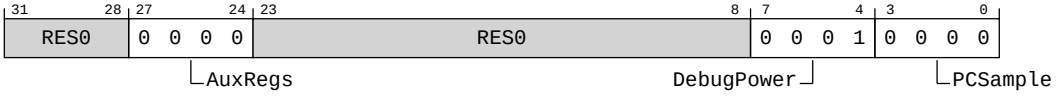


Table B-264: EDDEVID bit descriptions

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.4.31 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PE's debug logic.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

Register offset

0xFCC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0001 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-145: ext_eddevtype bit assignments

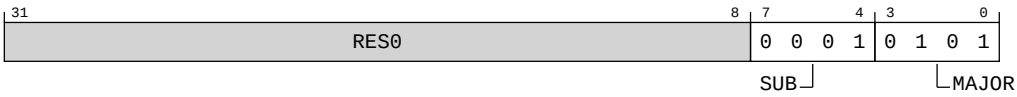


Table B-266: EDDEVTYPE bit descriptions

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.4.32 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFD0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-146: ext_edpidr4 bit assignments

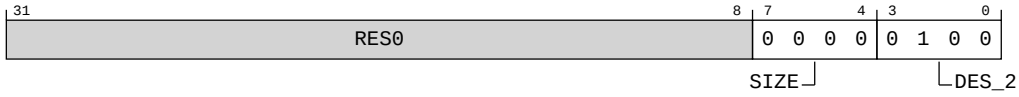


Table B-268: EDPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers. **0b0000**	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited. 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.4.33 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-147: ext_edpidr0 bit assignments

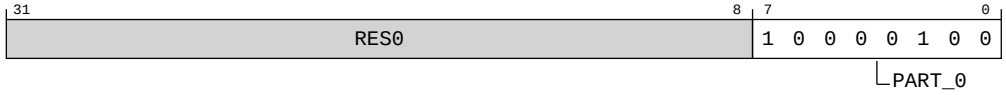


Table B-270: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PART_0	Part number, least significant byte. **0b10000100** Least Significant byte of the debug part number	0x84

Accessibility

Component	Offset	Instance	Range
Debug	0xFE0	EDPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.34 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-148: ext_edpidr1 bit assignments

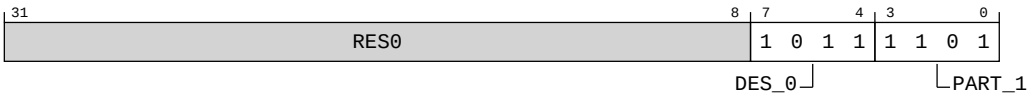


Table B-272: EDPIDR1 bit descriptions

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.4.35 EDPIDR2, External Debug Peripheral Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFE8

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-149: ext_edpidr2 bit assignments

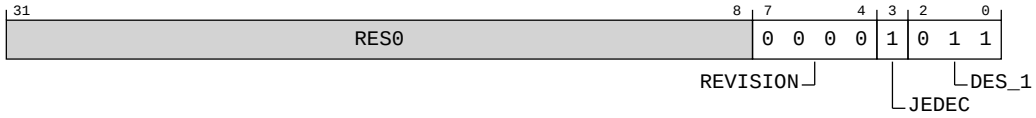


Table B-274: EDPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0000** r0p1	0b0000
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited. 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.4.36 EDPIDR3, External Debug Peripheral Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFEC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-150: ext_edpidr3 bit assignments

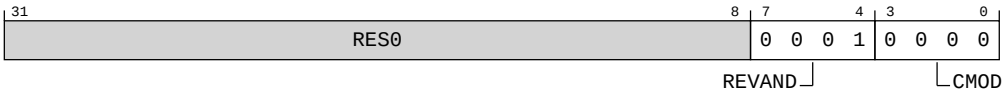


Table B-276: EDPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVAND	Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. **0b0001**	0b0001
[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.4.37 EDCIDR0, External Debug Component Identification Register 0

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width
32

Component
Debug

Register offset
0xFF0

Access type
See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-151: ext_edcidr0 bit assignments

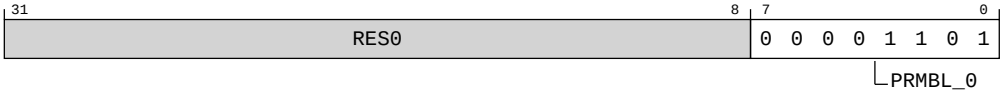


Table B-278: EDCIDR0 bit descriptions

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.4.38 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-152: ext_edcidr1 bit assignments

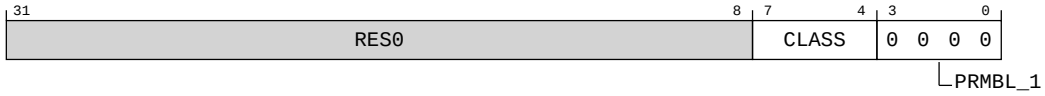


Table B-280: EDCIDR1 bit descriptions

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
Debug	0xFF4	EDCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.4.39 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFF8

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-153: ext_edcidr2 bit assignments

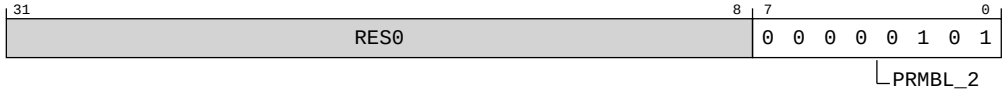


Table B-282: EDCIDR2 bit descriptions

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.4.40 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information, see *About the Component Identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

Debug

Register offset

0xFFC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-154: ext_edcidr3 bit assignments

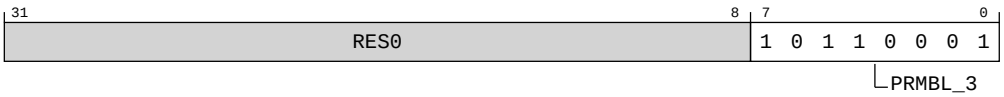


Table B-284: EDCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Preamble. **0b10110001**	0xB1

Accessibility

Component	Offset	Instance	Range
Debug	0xFFC	EDCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.5 External CTI registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table B-286: CTI registers summary

Offset	Name	Reset	Width	Description
0x150	CTIDEVCTL	See individual bit resets.	32-bit	CTI Device Control register
0xFE0	CTIPIDR0	See individual bit resets.	32-bit	CTI Peripheral Identification Register 0
0xFE4	CTIPIDR1	See individual bit resets.	32-bit	CTI Peripheral Identification Register 1

B.5.1 CTIPIDR0, CTI Peripheral Identification Register 0

Provides information to identify a CTI component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

CTI

Register offset

0xFE0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-155: ext_ctipidr0 bit assignments

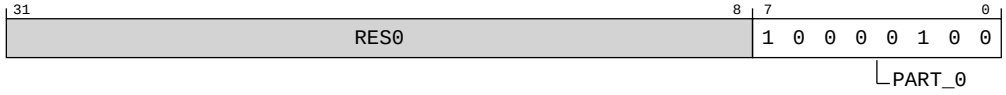


Table B-287: CTIPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PART_0	Part number, least significant byte. **0b10000100** Neoverse V3 Core ROM table. Bits [7:0] of part number 0xD84.	0x84

Accessibility

Component	Offset	Instance	Range
CTI	0xFE0	CTIPIDR0	None

This interface is accessible as follows:

RO

B.5.2 CTIPIDR1, CTI Peripheral Identification Register 1

Provides information to identify a CTI component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is required for CoreSight compliance.

Attributes

Width

32

Component

CTI

Register offset

0xFE4

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-156: ext_ctipidr1 bit assignments

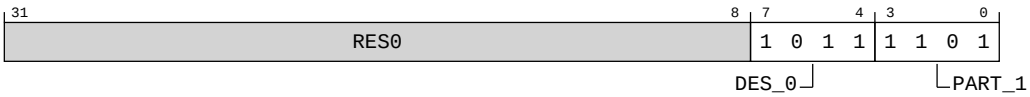


Table B-289: CTIPIDR1 bit descriptions

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`** Neoverse V3 Core ROM table. Bits [11:8] of part number 0xD84.	`0b1101`

Accessibility

Component	Offset	Instance	Range
CTI	0xFE4	CTIPIDR1	None

This interface is accessible as follows:

RO

B.6 External AMU registers summary

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

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

If a register does not have a link in the summary table, you can find more information about this Architecturally defined register in the .

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

Table B-291: AMU registers summary

Offset	Name	Reset	Width	Description
0x0	AMEVCNTR00 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x4	AMEVCNTR00 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0

Offset	Name	Reset	Width	Description
0x8	AMEVCNTR01 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0xC	AMEVCNTR01 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x10	AMEVCNTR02 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x14	AMEVCNTR02 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x18	AMEVCNTR03 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x1C	AMEVCNTR03 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 0
0x100	AMEVCNTR10 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x104	AMEVCNTR10 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x108	AMEVCNTR11 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x10C	AMEVCNTR11 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x110	AMEVCNTR12 [31:0]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x114	AMEVCNTR12 [63:32]	See individual bit resets.	32-bit	Activity Monitors Event Counter Registers 1
0x400	AMEVTYPER00	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	See individual bit resets.	32-bit	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	See individual bit resets.	32-bit	Activity Monitors Count Enable Clear Register 1
0xCE0	AMCGCR	See individual bit resets.	32-bit	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	See individual bit resets.	32-bit	Activity Monitors Configuration Register
0xE04	AMCR	See individual bit resets.	32-bit	Activity Monitors Control Register
0xE08	AMIIDR	See individual bit resets.	32-bit	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	See individual bit resets.	32-bit	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	See individual bit resets.	32-bit	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	See individual bit resets.	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	See individual bit resets.	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 3

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-157: ext_amevcntr00 bit assignments

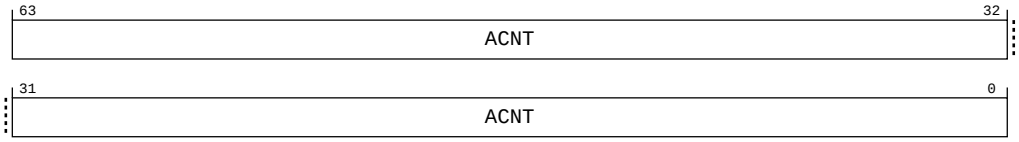


Table B-292: AMEVCNTR00 bit descriptions

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*.

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'.

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-158: ext_amevcntr01 bit assignments

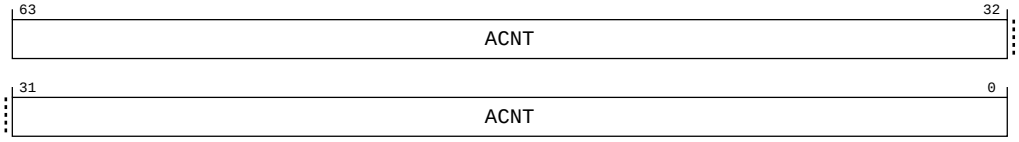


Table B-295: AMEVCNTR01 bit descriptions

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-159: ext_amevcntr02 bit assignments

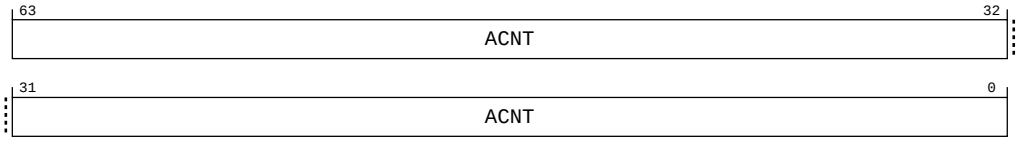


Table B-298: AMEVCNTR02 bit descriptions

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-160: ext_amevcntr03 bit assignments

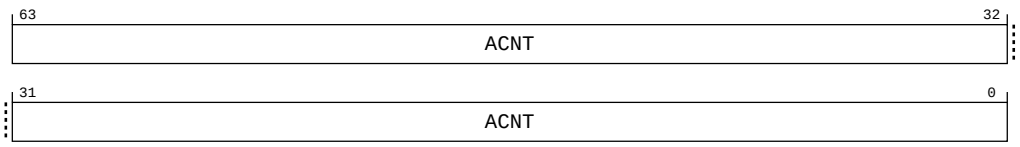


Table B-301: AMEVCNTR03 bit descriptions

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-161: ext_amevcntr10 bit assignments

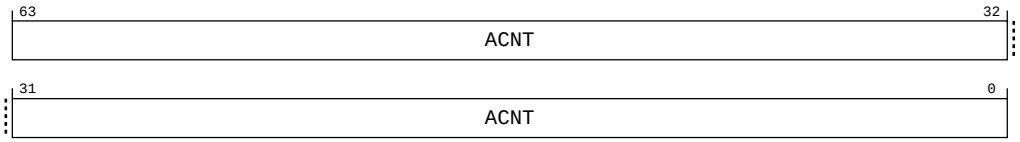


Table B-304: AMEVCNTR10 bit descriptions

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*.

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'.

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-162: ext_amevcntr11 bit assignments

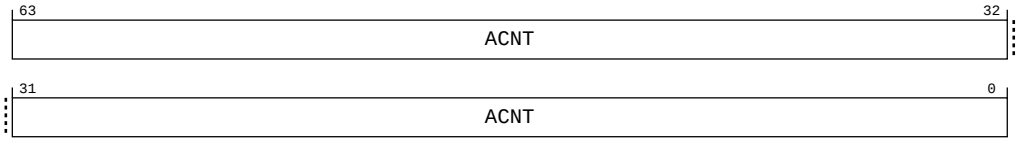


Table B-307: AMEVCNTR11 bit descriptions

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-163: ext_amevcntr12 bit assignments

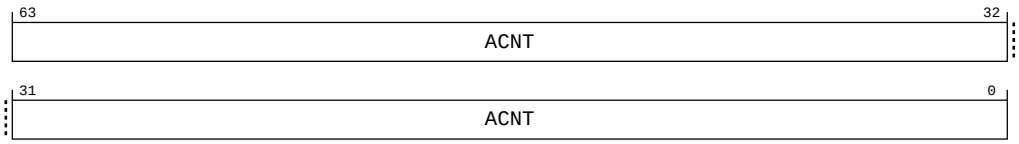


Table B-310: AMEVCNTR12 bit descriptions

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-164: ext_amevtyper00 bit assignments

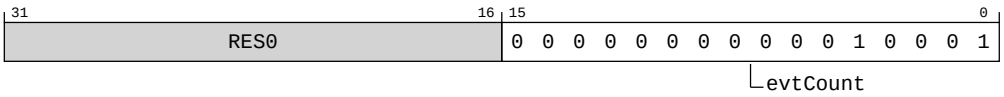


Table B-313: AMEVTYPER00 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0b0000000000010001** Processor frequency cycles	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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-165: ext_amevtyper01 bit assignments

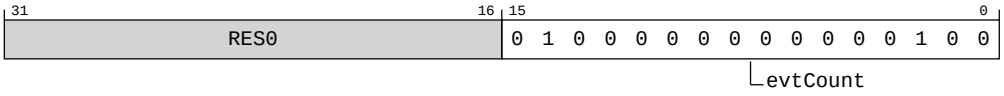


Table B-315: AMEVTYPER01 bit descriptions

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: **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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-166: ext_amevtyper02 bit assignments

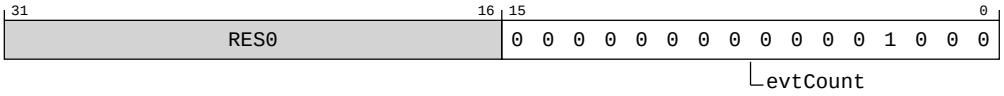


Table B-317: AMEVTYPER02 bit descriptions

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*.

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'.

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-167: ext_amevtyper03 bit assignments

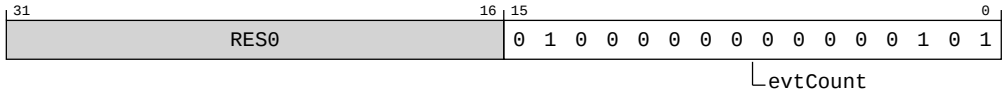


Table B-319: AMEVTYPER03 bit descriptions

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-168: ext_amevtyper10 bit assignments

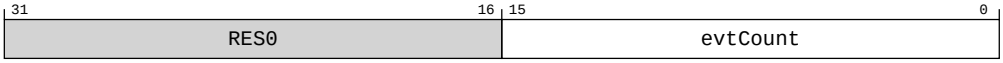


Table B-321: AMEVTYPER10 bit descriptions

Bits	Name	Description	Reset
[31:16]	**RES0**	Reserved	**RES0**
[15:0]	evtCount	Event to count. The event number of the event that is counted by the 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 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-169: ext_amevtyper11 bit assignments

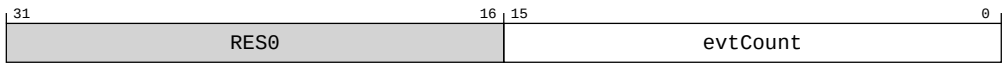


Table B-323: AMEVTYPER11 bit descriptions

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 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 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-170: ext_amevtyper12 bit assignments

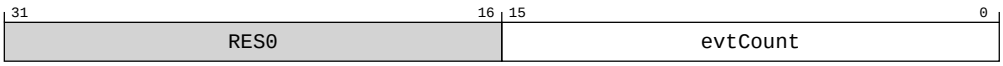


Table B-325: AMEVTYPER12 bit descriptions

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 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 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 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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-171: ext_amcgcr bit assignments

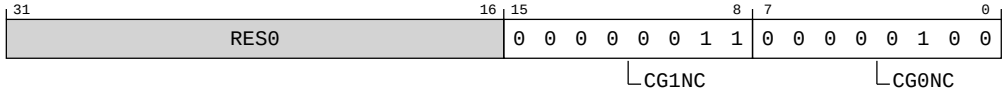


Table B-327: AMCGCR bit descriptions

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.6.16 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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-172: ext_amcfgr bit assignments

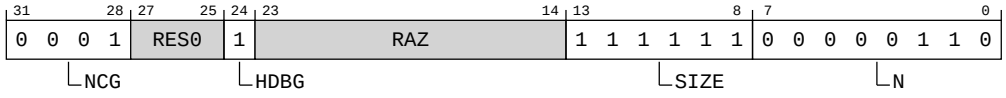


Table B-329: AMCFGR bit descriptions

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**

Bits	Name	Description	Reset
[13:8]	SIZE	Defines the size of activity monitor event counters. The size of the activity monitor event counters implemented by the Activity Monitors Extension is [AMCFGR.SIZE + 1]. The counters are 64-bit. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. The counters are at doubleword-aligned addresses. **0b111111**	0b111111
[7:0]	N	Defines the number of activity monitor event counters. The total number of counters implemented in all groups by the Activity Monitors Extension is [AMCFGR.N + 1]. **0b00000110** Seven activity monitor event counters	0x06

Accessibility

Component	Offset	Instance	Range
AMU	0xE00	AMCFGR	None

This interface is accessible as follows:

RO

B.6.17 AMIIDR, Activity Monitors Implementation Identification Register

Defines the implementer and revisions of the AMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xE08

Access type

RO

Reset value

```
1101 1000 0100 0000 0001 xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-173: ext_amiidr bit assignments

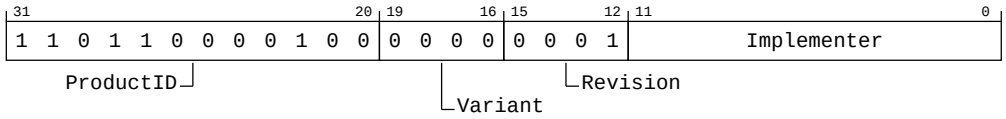


Table B-331: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. **0b110110000100** Neoverse V3	0xD84
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. **0b0000** r0p1 If ext-AMPIDR2 is implemented, ext-AMPIDR2.REVISION matches AMIIDR.Variant.	0b0000
[15:12]	Revision	This field distinguishes minor revisions of the product. **0b0001** r0p1 If ext-AMPIDR3 is implemented, ext-AMPIDR3.REVAND matches AMIIDR.Revision.	0b0001
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the AMU. For an Arm implementation, this field reads as 0x43B.	12{x}

Accessibility

Component	Offset	Instance	Range
AMU	0xE08	AMIIDR	None

This interface is accessible as follows:

RO

B.6.18 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-174: ext_amdevarch bit assignments

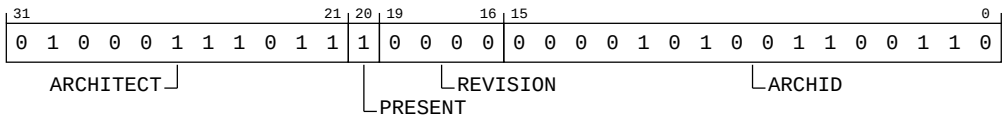


Table B-333: AMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For AMU, this is Arm Limited. **0b01000111011**	0b01000111011
[20]	PRESENT	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.	0b0000

Bits	Name	Description	Reset
[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.6.19 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-175: ext_amdevtype bit assignments

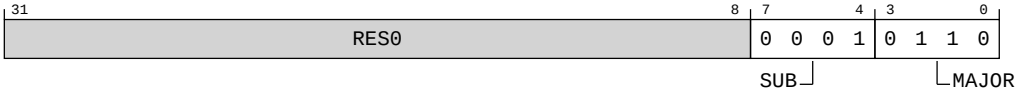


Table B-335: AMDEVTYPE bit descriptions

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.6.20 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-176: ext_ampidr4 bit assignments

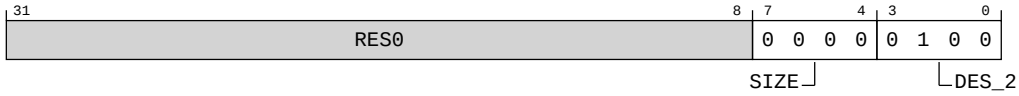


Table B-337: AMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	SIZE	Size of the component. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers. **0b0000**	0b0000
[3:0]	DES_2	Designer. JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **0b0100** Arm Limited. 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.6.21 AMPIDR0, Activity Monitors Peripheral Identification Register 0

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-177: ext_ampidr0 bit assignments

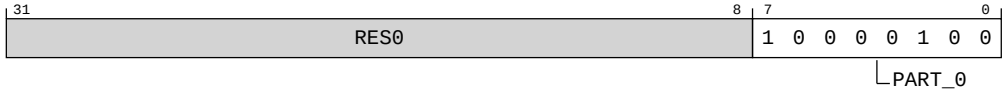


Table B-339: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PART_0	Part number, least significant byte. **0b10000100** Part number, least significant byte.	0x84

Accessibility

Component	Offset	Instance	Range
AMU	0xFE0	AMPIDR0	None

This interface is accessible as follows:

RO

B.6.22 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-178: ext_ampidr1 bit assignments

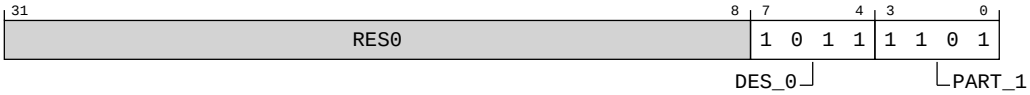


Table B-341: AMPIDR1 bit descriptions

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.6.23 AMPIDR2, Activity Monitors Peripheral Identification Register 2

Provides information to identify an activity monitors component.

For more information, see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

AMU

Register offset

0xFE8

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-179: ext_ampidr2 bit assignments

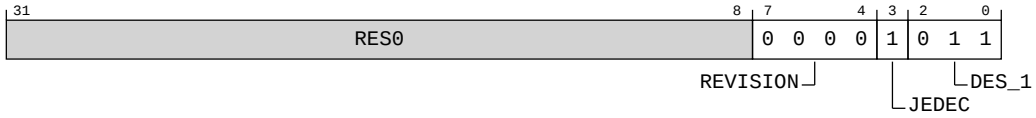


Table B-343: AMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. **0b0000** r0p1	0b0000
[3]	JEDEC	Indicates a JEP106 identity code is used. **0b1**	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. **0b011** Arm Limited. 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.6.24 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 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-180: ext_ampidr3 bit assignments

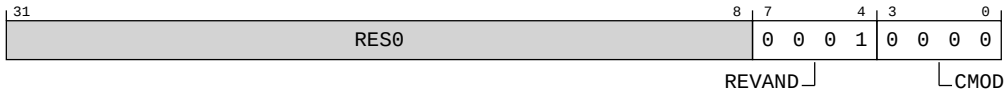


Table B-345: AMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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. **0b0001**	0b0001
[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.6.25 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-181: ext_amcidr0 bit assignments

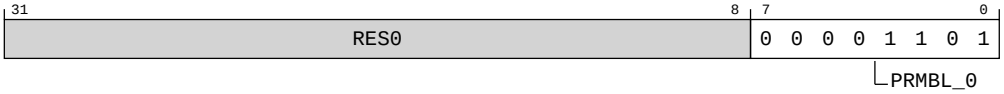


Table B-347: AMCIDR0 bit descriptions

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.6.26 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-182: ext_amcidr1 bit assignments

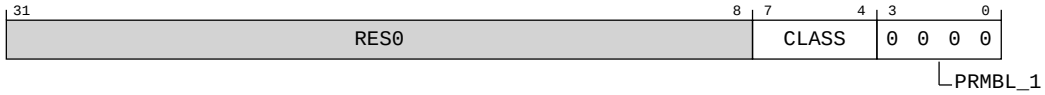


Table B-349: AMCIDR1 bit descriptions

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.6.27 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-183: ext_amcidr2 bit assignments

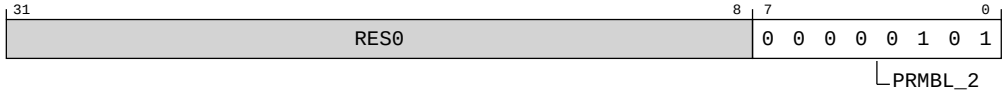


Table B-351: AMCIDR2 bit descriptions

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.6.28 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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-184: ext_amcidr3 bit assignments

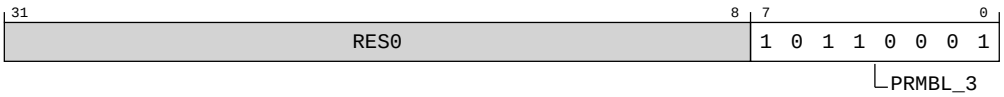


Table B-353: AMCIDR3 bit descriptions

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

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

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

Table B-355: ETE registers summary

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	See individual bit resets.	32-bit	Auxiliary Control Register
0x180	TRCIDR8	See individual bit resets.	32-bit	ID Register 8
0x184	TRCIDR9	See individual bit resets.	32-bit	ID Register 9
0x188	TRCIDR10	See individual bit resets.	32-bit	ID Register 10
0x18C	TRCIDR11	See individual bit resets.	32-bit	ID Register 11
0x190	TRCIDR12	See individual bit resets.	32-bit	ID Register 12
0x194	TRCIDR13	See individual bit resets.	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	See individual bit resets.	32-bit	IMP DEF Register 0

Offset	Name	Reset	Width	Description
0x1E0	TRCIDR0	See individual bit resets.	32-bit	ID Register 0
0x1E4	TRCIDR1	See individual bit resets.	32-bit	ID Register 1
0x1E8	TRCIDR2	See individual bit resets.	32-bit	ID Register 2
0x1EC	TRCIDR3	See individual bit resets.	32-bit	ID Register 3
0x1F0	TRCIDR4	See individual bit resets.	32-bit	ID Register 4
0x1F4	TRCIDR5	See individual bit resets.	32-bit	ID Register 5
0x1F8	TRCIDR6	See individual bit resets.	32-bit	ID Register 6
0x1FC	TRCIDR7	See individual bit resets.	32-bit	ID Register 7
0xF00	TRCITCTRL	See individual bit resets.	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	See individual bit resets.	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	See individual bit resets.	32-bit	Claim Tag Clear Register
0xFBC	TRCDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	See individual bit resets.	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	See individual bit resets.	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	See individual bit resets.	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	See individual bit resets.	32-bit	Device Type Register
0xFD0	TRCPIDR4	See individual bit resets.	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	See individual bit resets.	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	See individual bit resets.	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	See individual bit resets.	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	See individual bit resets.	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	See individual bit resets.	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	See individual bit resets.	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	See individual bit resets.	32-bit	Component Identification Register 0
0xFF4	TRCCIDR1	See individual bit resets.	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	See individual bit resets.	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	See individual bit resets.	32-bit	Component Identification Register 3

B.7.1 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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-185: ext_trcauxctlr bit assignments

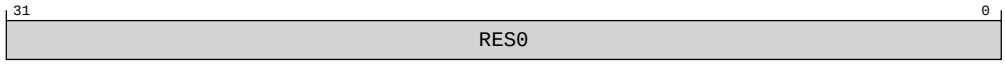


Table B-356: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

If this register is nonzero then it might cause the behavior of a trace unit to contradict this architecture specification. See the documentation of the specific implementation for information about the IMPLEMENTATION DEFINED support for this register.

Component	Offset	Instance	Range
ETE	0x018	TRCAUXCTLR	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.2 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-186: ext_trcidr8 bit assignments

Table B-358: 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.7.3 TRCIDR9, ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x184

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-187: ext_trcidr9 bit assignments

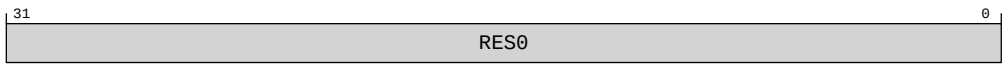


Table B-360: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
ETE	0x184	TRCIDR9	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.4 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-188: ext_trcidr10 bit assignments

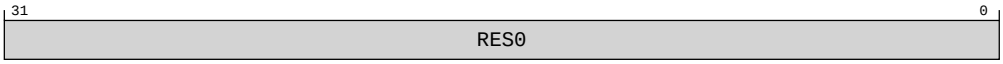


Table B-362: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x188	TRCIDR10	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.5 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-189: ext_trcidr11 bit assignments

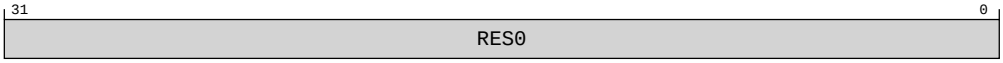


Table B-364: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x18C	TRCIDR11	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.6 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-190: ext_trcidr12 bit assignments

Table B-366: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x190	TRCIDR12	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.7 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-191: ext_trcidr13 bit assignments

Table B-368: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x194	TRCIDR13	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.8 TRCIMSPEC0, IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-192: ext_trcimspec0 bit assignments

Table B-370: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[31:4]	**RES0**	Reserved	**RES0**
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0b0000** No **IMPLEMENTATION DEFINED** features are supported.	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1C0	TRCIMSPEC0	None

This interface is accessible as follows:

When OSLockStatus() || !AllowExternalTraceAccess() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.9 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-193: ext_trcidr0 bit assignments

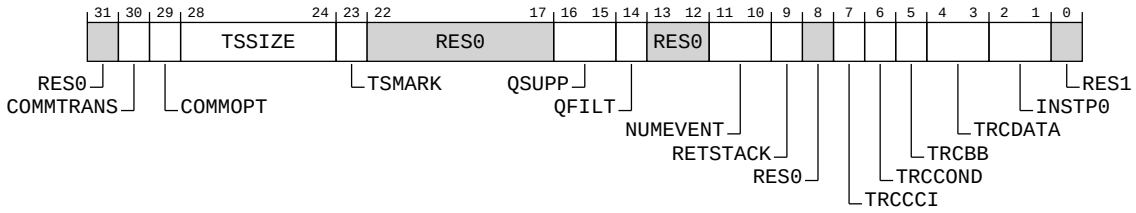


Table B-372: TRCIDR0 bit descriptions

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]	TSMARK	Indicates whether Timestamp Marker elements are generated. **0b0** Timestamp Marker elements are not generated. **0b1** Timestamp Marker elements are generated.	x
[22:17]	**RES0**	Reserved	**RES0**
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **0b00** Q element support is not implemented.	xx
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0b0** Q element filtering is not implemented.	x
[13:12]	**RES0**	Reserved	**RES0**
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **0b11** The trace unit supports 4 ETEEvents.	xx

Bits	Name	Description	Reset
[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.7.10 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1E4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-194: ext_trcidr1 bit assignments

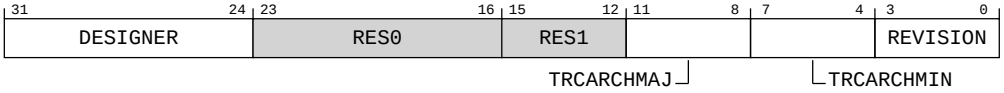


Table B-374: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. **0b01000001** Arm Limited	8{x}
[23:16]	**RES0**	Reserved	**RES0**
[15:12]	**RES1**	Reserved	**RES1**
[11:8]	TRCARCHMAJ	Major architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	xxxx
[7:4]	TRCARCHMIN	Minor architecture version. **0b1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	xxxx

Bits	Name	Description	Reset
[3:0]	REVISION	Implementation revision that identifies the revision of the trace and OS Lock registers. **0b0000** Revision 0	xxxx

Accessibility

Component	Offset	Instance	Range
ETE	0x1E4	TRCIDR1	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.11 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-195: ext_trcidr2 bit assignments

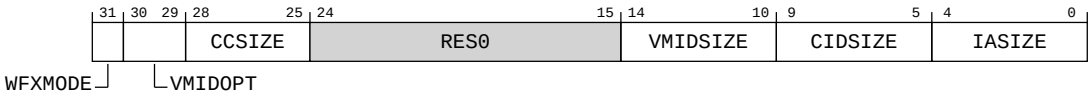


Table B-376: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions: **0b1** `WFI`, `WFIT`, `WFE`, and `WFET` instructions are classified as P0 instructions.	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.7.12 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1EC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-196: ext_trcidr3 bit assignments

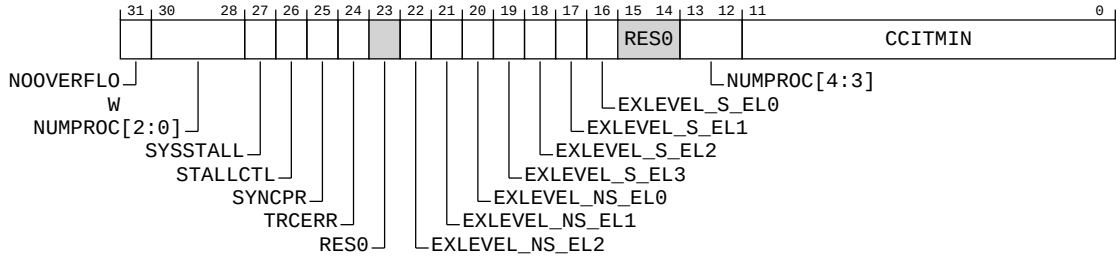


Table B-378: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0b0** Overflow prevention is not implemented.	x

Bits	Name	Description	Reset
[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**
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **0b00000** The trace unit can trace one PE.	5{x}

Bits	Name	Description	Reset
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in ext-TRCCCCTLR.THRESHOLD. If ext-TRCIDR0.TRCCCI == 1 then the minimum value of this field is `0x001`. If ext-TRCIDR0.TRCCCI == 0 then this field is zero.	`12{x}`

Accessibility

Component	Offset	Instance	Range
ETE	0x1EC	TRCIDR3	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.13 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-197: ext_trcidr4 bit assignments

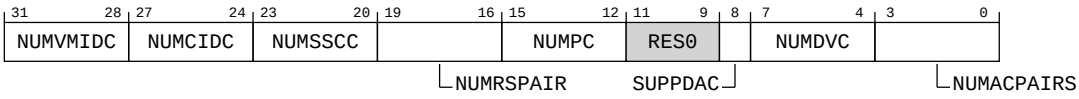


Table B-380: TRCIDR4 bit descriptions

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.7.14 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0x1F4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-198: ext_trcidr5 bit assignments

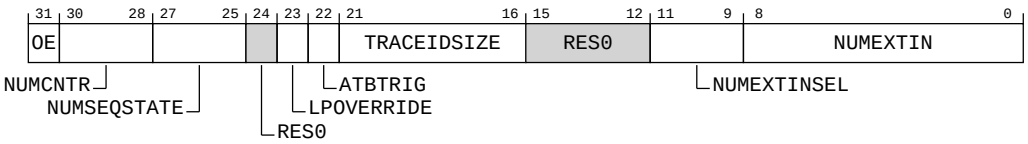


Table B-382: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[31]	OE	Indicates support for the ETE Trace Output Enable. **0b0** ETE Trace Output Enable is not implemented. **0b1** ETE Trace Output Enable is implemented.	The reset values can be the following: `0b0`, `0b1`, respective to the value.
[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.	`9{x}`

Accessibility

Component	Offset	Instance	Range
ETE	0x1F4	TRCIDR5	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.15 TRCIDR6, ID Register 6

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-199: ext_trcidr6 bit assignments

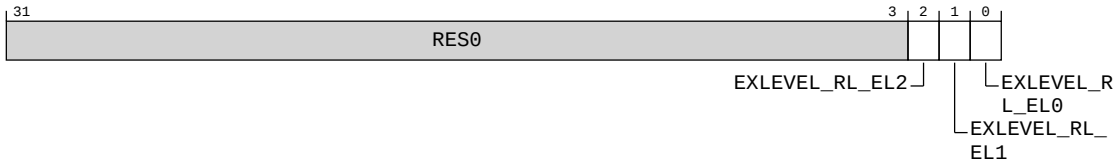


Table B-384: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:3]	**RES0**	Reserved	**RES0**
[2]	EXLEVEL_RL_EL2	Indicates if Realm EL2 is implemented. **0b1** Realm EL2 is implemented.	x

Bits	Name	Description	Reset
[1]	EXLEVEL_RL_EL1	Indicates if Realm EL1 is implemented. **0b1** Realm EL1 is implemented.	x
[0]	EXLEVEL_RL_EL0	Indicates if Realm EL0 is implemented. **0b1** Realm EL0 is implemented.	x

Accessibility

Component	Offset	Instance	Range
ETE	0x1F8	TRCIDR6	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RO

B.7.16 TRCIDR7, ID Register 7

Returns the tracing capabilities of the trace unit.

Configurations

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
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-200: ext_trcidr7 bit assignments

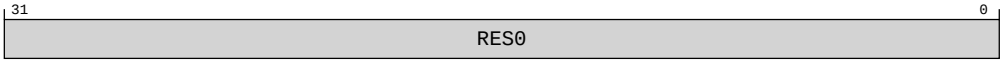


Table B-386: TRCIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	**RES0**	Reserved	**RES0**

Accessibility

Component	Offset	Instance	Range
ETE	0x1FC	TRCIDR7	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.17 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
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-201: ext_trcitctrl bit assignments

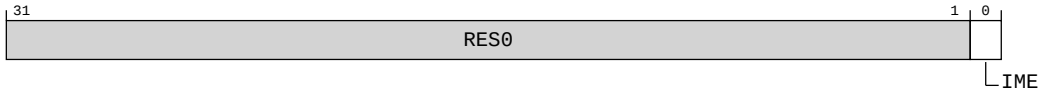


Table B-388: TRCITCTRL bit descriptions

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.7.18 TRCCLAIMSET, Claim Tag Set Register

In conjunction with ext-TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

The number of claim tag bits implemented is IMPLEMENTATION DEFINED. Arm recommends that implementations support a minimum of four claim tag bits, that is, SET[3:0] reads as 0b1111.

Attributes

Width

32

Component

ETE

Register offset

0xFA0

Access type

RAOW1S

Reset value

1111 1111 1111 1111 1111 1111 1111 1111

Bit descriptions

Figure B-202: ext_trcclaimset bit assignments

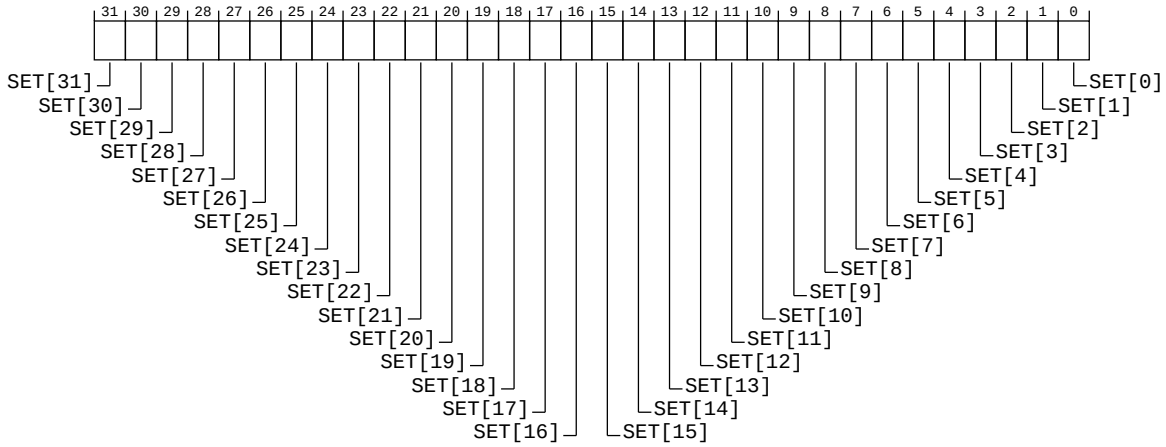


Table B-390: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:0]	SET[<m>], bit[m], where m = 31 to 0	Claim Tag Set. Indicates whether Claim Tag bit <m> is implemented, and is used to set Claim Tag bit <m> to 1. **0b0** On a read: Claim Tag bit <m> is not implemented. On a write: Ignored. **0b1** On a read: Claim Tag bit <m> is implemented. On a write: Set Claim Tag bit <m> to 1.	0xFFFFFFFF

Accessibility

Component	Offset	Instance	Range
ETE	0xFA0	TRCCLAIMSET	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()
ERROR

Otherwise
RW

B.7.19 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with ext-TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFA4

Access type
RW1C

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-203: ext_trcclaimclr bit assignments

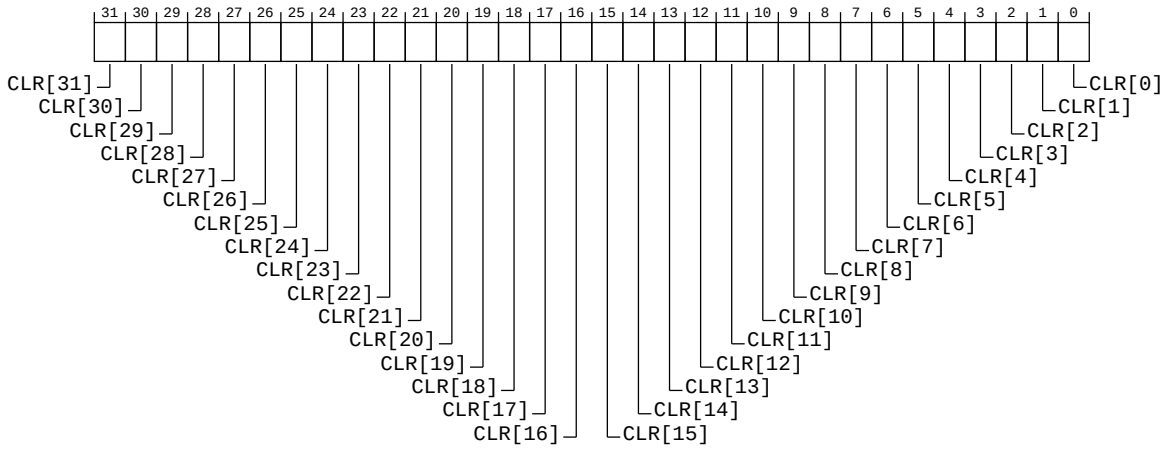


Table B-392: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:0]	CLR[<m>], bit[m], where m = 31 to 0	Claim Tag Clear. Indicates the current status of Claim Tag bit <m>, and is used to clear Claim Tag bit <m> to 0. **0b0** On a read: Claim Tag bit <m> is not set. On a write: Ignored. **0b1** On a read: Claim Tag bit <m> is set. On a write: Clear Claim tag bit <m> to 0.	0x00000000

Accessibility

Component	Offset	Instance	Range
ETE	0xFA4	TRCCLAIMCLR	None

This interface is accessible as follows:

When OSLockStatus() || !IsTraceCorePowered()

ERROR

Otherwise

RW

B.7.20 TRCDEVARCH, Device Architecture Register

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFBC

Access type

See bit descriptions

Reset value

0100 0111 0111 0010 0101 1010 0001 0011

Bit descriptions

Figure B-204: ext_trcdevarch bit assignments

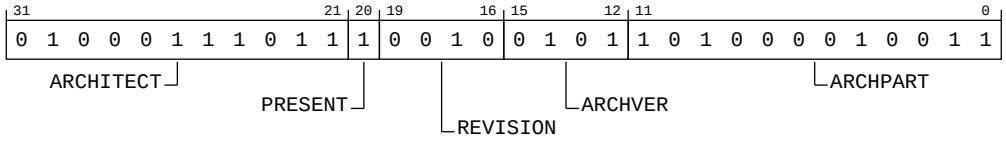


Table B-394: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. **0b01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	0b01000111011
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **0b1** Device Architecture information present.	0b1
[19:16]	REVISION	Revision. Defines the architecture revision of the component. Defined values are: **0b0010** ETEv1.2, FEAT_ETEv1p2.	0b0010

Bits	Name	Description	Reset
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0b0101** ETEv1.	0b0101
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **0b101000010011** Arm PE trace architecture.	0xA13

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFBC	TRCDEVARCH	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.7.21 TRCDEVID2, Device Configuration Register 2

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-205: ext_trcdevid2 bit assignments

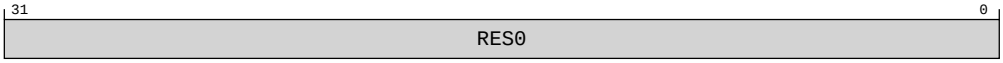


Table B-396: TRCDEVID2 bit descriptions

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.7.22 TRCDEVID1, Device Configuration Register 1

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-206: ext_trcdevid1 bit assignments

Table B-398: 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.7.23 TRCDEVID, Device Configuration Register

Provides discovery information for the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFC8

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-207: ext_trcdevid bit assignments

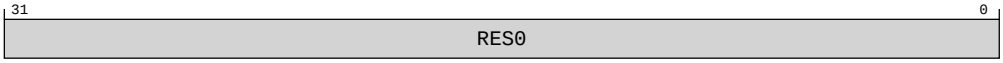


Table B-400: TRCDEVID bit descriptions

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.7.24 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFCC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-208: ext_trcdevtype bit assignments

Table B-402: 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.	xxxx
[3:0]	MAJOR	Component major type. **0b0011** Trace source.	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.7.25 TRCPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFD0

Access type
See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100
```

```
|    |    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3    0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-209: ext_trcpidr4 bit assignments

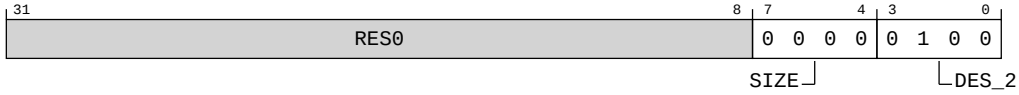


Table B-404: TRCPIDR4 bit descriptions

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`**	`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.	`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.7.26 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFD4

Access type
See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-210: ext_trcpidr5 bit assignments

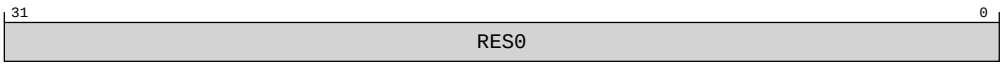


Table B-406: TRCPIDR5 bit descriptions

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.7.27 TRCPIDR6, Peripheral Identification Register 6

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFD8

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-211: ext_trcpidr6 bit assignments

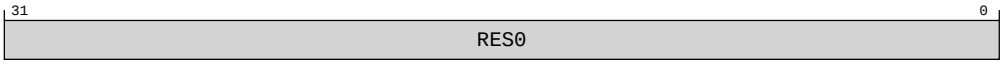


Table B-408: TRCPIDR6 bit descriptions

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.7.28 TRCPIDR7, Peripheral Identification Register 7

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFDC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-212: ext_trcpidr7 bit assignments

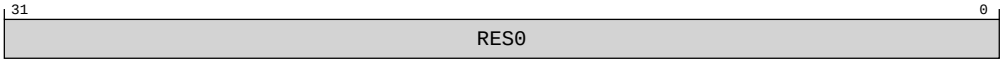


Table B-410: TRCPIDR7 bit descriptions

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.7.29 TRCPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1000 0100
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-213: ext_trcpidr0 bit assignments

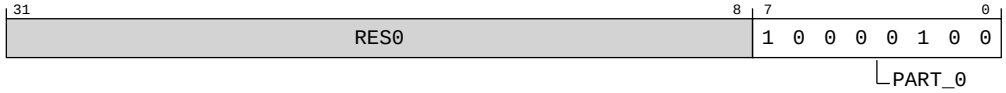


Table B-412: TRCPIDR0 bit descriptions

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. **0b10000100** Least significant byte of the ETM trace unit part.	0x84

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.7.30 TRCPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1011 1101
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-214: ext_trcpidr1 bit assignments

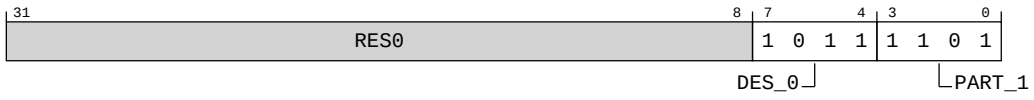


Table B-414: TRCPIDR1 bit descriptions

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.	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.7.31 TRCPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFE8

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 1011
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-215: ext_trcpidr2 bit assignments

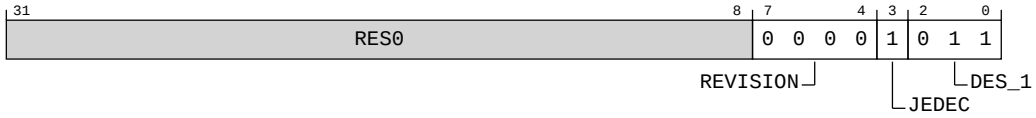


Table B-416: TRCPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:4]	REVISION	Component major revision. TRCPIDR2.REVISION and ext-TRCPIDR3.REVAND together form the revision number of the component, with TRCPIDR2.REVISION being the most significant part and ext-TRCPIDR3.REVAND the least significant part. When a component is changed, TRCPIDR2.REVISION or ext-TRCPIDR3.REVAND are increased to ensure that software can differentiate the different revisions of the component. ext-TRCPIDR3.REVAND should be set to `0b0000` when TRCPIDR2.REVISION is increased. **`0b0000`** r0p1 - Part major revision.	`0b0000`
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. **`0b1`**	`0b1`
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. ext-TRCPIDR1.DES_0 and TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. **`0b011`** Arm Limited. Most significant nibble of JEP106 ID code.	`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.7.32 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFEC

Access type
See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0001 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-216: ext_trcpidr3 bit assignments

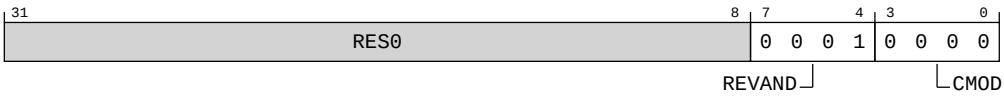


Table B-418: TRCPIDR3 bit descriptions

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. **`0b0001`** Part minor revision.	`0b0001`
[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.	`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.7.33 TRCCIDR0, Component Identification Register 0

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 1101
|    |    |    |    |    |    |    | |
31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-217: ext_trccidr0 bit assignments

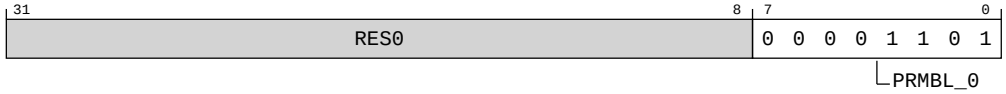


Table B-420: TRCCIDR0 bit descriptions

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.7.34 TRCCIDR1, Component Identification Register 1

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFF4

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-218: ext_trccidr1 bit assignments

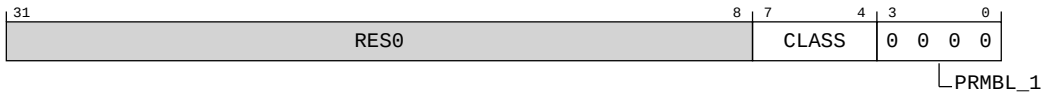


Table B-422: TRCCIDR1 bit descriptions

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.7.35 TRCCIDR2, Component Identification Register 2

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETE

Register offset
0xFF8

Access type
See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 0101
|    |    |    |    |    |    |    | |
```

```
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-219: ext_trccidr2 bit assignments

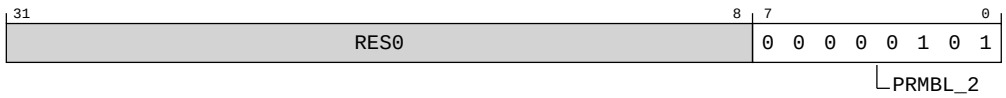


Table B-424: TRCCIDR2 bit descriptions

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.7.36 TRCCIDR3, Component Identification Register 3

Provides discovery information about the component.

For additional information, see the CoreSight Architecture Specification.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETE

Register offset

0xFFC

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1011 0001
|    |    |    |    |    |    |    |  |
31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-220: ext_trccidr3 bit assignments

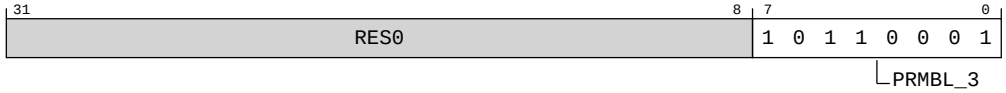


Table B-426: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	**RES0**	Reserved	**RES0**
[7:0]	PRMBL_3	Component identification preamble, segment 3. **0b10110001**	0xB1

Accessibility

External debugger accesses to this register are unaffected by the OS Lock.

Component	Offset	Instance	Range
ETE	0xFFC	TRCCIDR3	None

This interface is accessible as follows:

When !IsTraceCorePowered()

ERROR

Otherwise

RO

B.8 External RAS registers summary

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

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

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

Table B-428: RAS registers summary

Offset	Name	Reset	Width	Description
0x0	ERR0FR	See individual bit resets.	64-bit	Error Record <n> Feature Register
0x8	ERR0CTLR	See individual bit resets.	64-bit	Error Record <n> Control Register
0x10	ERR0STATUS	See individual bit resets.	64-bit	Error Record <n> Primary Status Register
0x18	ERR0ADDR	See individual bit resets.	64-bit	Error Record <n> Address Register
0x20	ERR0MISC0	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 0
0x28	ERR0MISC1	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 1
0x30	ERR0MISC2	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 2
0x38	ERR0MISC3	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 3
0x800	ERR0PFGF	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Feature Register
0x808	ERR0PFGCTL	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Control Register
0x810	ERR0PFGCDN	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Countdown Register

B.8.1 ERR0FR, Error Record <n> Feature Register

Defines whether error record <n> is the first record owned by a node:

- If error record <n> is the first error record owned by a node, then ERR<n>FR.ED is not 0b00.
- If error record <n> is not the first error record owned by a node, then ERR<n>FR.ED is 0b00.

If error record <n> is the first record owned by the node, defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

RAS

Register offset

0x0

Access type

RO

Reset value

```
xxxx xxxx x101 0001 xxxx xxxx xxxx xxxx 1xxx xx00 0001 0000 1010 1001 1010 xx10
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-221: ext_err0fr bit assignments

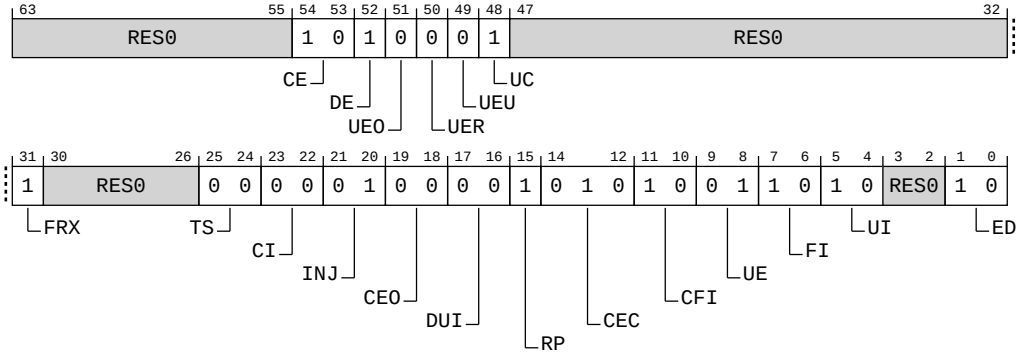


Table B-429: ERR0FR bit descriptions

Bits	Name	Description	Reset
[63:55]	**RES0**	Reserved	**RES0**
[54:53]	CE	Corrected Error recording. Describes the types of Corrected errors the node can record, if any. **0b10** Records only non-specific Corrected errors. That is, Corrected errors recorded by setting ERXSTATUS_EL1.CE to 0b10.	0b10

Bits	Name	Description	Reset
[52]	DE	Deferred Error recording. Describes whether the node supports recording Deferred errors. **0b1** Records Deferred errors.	0b1
[51]	UEO	Latent or Restartable Error recording. Describes whether the node supports recording Latent or Restartable errors. **0b0** Does not record Latent or Restartable errors.	0b0
[50]	UER	Signaled or Recoverable Error recording. Describes whether the node supports recording Signaled or Recoverable errors. **0b0** Does not record Signaled or Recoverable errors.	0b0
[49]	UEU	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b0** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b0
[48]	UC	Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors. **0b1** Uncontainable Error recording. Describes whether the node supports recording Uncontainable errors.	0b1
[47:32]	**RES0**	Reserved	**RES0**
[31]	FRX	Feature Register extension. Defines whether ERXFR_EL1[63:48] are architecturally defined. **0b1** ERXFR_EL1[63:48] are defined by the architecture.	0b1
[30:26]	**RES0**	Reserved	**RES0**
[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

Bits	Name	Description	Reset
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. **0b1** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	0b1
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in ERXMISC0_EL1 for each error record <m> owned by the node that can record countable errors. **0b010** Implements an 8-bit Corrected error counter in ERXMISC0_EL1[39:32].	0b010
[11:10]	CFI	Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented. **0b10** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI.	0b10
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. **0b01** In-band uncorrected error reporting (External Aborts) is supported and always enabled. ERXCTLR_EL1.UE is **RES0**.	0b01
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. **0b10** Fault handling interrupt is supported and controllable using ERXCTLR_EL1.FI.	0b10
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. **0b10** Error handling interrupt is supported and controllable using ERXCTLR_EL1.UI.	0b10
[3:2]	**RES0**	Reserved	**RES0**
[1:0]	ED	Error reporting and logging. Indicates whether error record <n> is the first record owned the node, and, if so, whether it implements the controls for enabling and disabling error reporting and logging. **0b10** Error reporting and logging is controllable using ERXCTLR_EL1.ED.	0b10

Accessibility

Component	Offset	Instance	Range
RAS	0x0	ERR0FR	None

This interface is accessible as follows:

RO

B.8.2 ERR0CTLR, Error Record <n> Control Register

The error control register contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling the critical error, error recovery, and fault handling interrupts.
- Enabling in-band error response for uncorrected errors.

For each bit, if the node does not support the feature, then the bit is **RES0**. The definition of each record is IMPLEMENTATION DEFINED.

Configurations

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x8

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 xxxx 00x0
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-222: ext_err0ctlr bit assignments

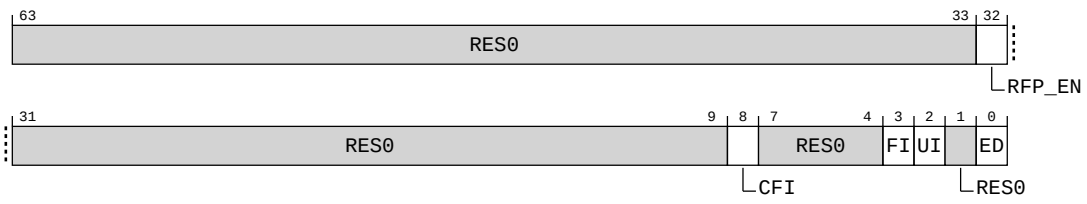


Table B-431: ERR0CTLR bit descriptions

Bits	Name	Description	Reset
[63:33]	**RES0**	Reserved	**RES0**
[32]	RFP_EN	Register File Parity error reporting and logging enable. When disabled, the node behaves as if Register File Parity error detection is disabled, and no RFP errors are recorded or signaled by the core. This control applies to Register File Parity errors when error logging and reporting is enabled via ext-ERR<n>CTRL.ED. **0b0** Register File Parity error reporting disabled. **0b1** Register File Parity error reporting enabled.	x
[31:9]	**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**
[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

Bits	Name	Description	Reset
[2]	UI	Uncorrected error recovery interrupt enable. This control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. **0b0** Error recovery interrupt disabled. **0b1** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error. Cold reset only. Unaffected by Warm reset	0b0
[1]	**RES0**	Reserved	**RES0**
[0]	ED	Error Detection and correction enable. The possible values are: **0b0** Error detection and correction disabled. **0b1** Error detection and correction enabled. Cold reset only. Unaffected by Warm reset	0b0

Accessibility

Component	Offset	Instance	Range
RAS	0x8	ERR0CTLR	None

This interface is accessible as follows:

RW

B.8.3 ERR0STATUS, Error Record <n> Primary Status Register

Contains status information for error record <n>, including:

- Whether any error has been detected (valid).
- Whether any detected error was not corrected, and returned to a Requester.
- Whether any detected error was not corrected and deferred.
- Whether an error record has been discarded because additional errors have been detected before the first error was handled by software (overflow).
- Whether any error has been reported.
- Whether the other error record registers contain valid information.

- Whether the error was reported because poison data was detected or because a corrupt value was detected by an error detection code.
- A primary error code.
- An **IMPLEMENTATION DEFINED** extended error code.

Within this register:

- ERR<n>STATUS.{AV, V, MV} are valid bits that define whether error record <n> registers are valid.
- ERR<n>STATUS.{UE, OF, CE, DE, UET} encode the types of error or errors recorded.
- ERR<n>STATUS.{CI, ER, PN, IERR, SERR} are syndrome fields.

Configurations

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x10

Access type

Read

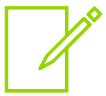
R

Write

W

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000 0000 0000 xxxx xxxx xxxx xxx0 0000
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-223: ext_err0status bit assignments

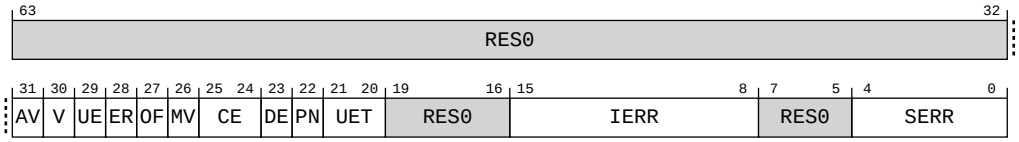


Table B-433: ERR0STATUS bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[31]	AV	Address Valid. The possible values are: **0b0** ERXADDR_EL1 not valid. **0b1** ERXADDR_EL1 contains an address associated with the highest priority error recorded by this record. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[30]	V	Status Register Valid. The possible values are: **0b0** ERXSTATUS_EL1 not valid. **0b1** ERXSTATUS_EL1 valid. At least one error has been recorded. This bit is read/write-one-to-clear. Cold reset only. Unaffected by Warm reset	0b0
[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
[19:16]	**RES0**	Reserved	**RES0**
[15:8]	IERR	Used with primary error code SERR value for additional information about the error. The core uses IERR values in conjunction with SERR code 0b11010 to provide the source of internal state parity error if identifiable. When other SERR codes are valid, IERR information may be stale or invalid and shall not be used. The possible values are: **0b00000000** Unknown source of internal state parity error **0b00000001** Parity error on General Register File **0b00000010** Parity error on Vector Register File	8{x}
[7:5]	**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. **0b11010** Parity error on internal state of the core including register files protected via Register File Parity Cold reset only. Unaffected by Warm reset	0b00000

Access

ERR<n>STATUS.{AV, V, UE, ER, OF, MV, CE, DE, PN, UET, CI} are write-one-to-clear (W1C) fields, meaning writes of zero are ignored, and a write of one or all-ones to the field clears the field to zero. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is **IMPLEMENTATION DEFINED**. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, software must clear the valid fields in the register to allow new errors to be recorded. However, between reading the register and clearing the valid fields, a new error might have overwritten the register. To prevent this error being lost by software, the register prevents updates to fields that might have been updated by a new error.

When RAS System Architecture v1.0 is implemented:

- Writes to ERR<n>STATUS.{UE, DE, CE} are ignored if ERR<n>STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR<n>STATUS.V are ignored if any of ERR<n>STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR<n>STATUS.{AV, MV} and the ERR<n>STATUS.{ER, PN, UET, IERR, SERR} syndrome fields are ignored if the highest priority nonzero error status field is not being cleared to zero. The error status fields in priority order from highest to lowest, are ERR<n>STATUS.UE, ERR<n>STATUS.DE, and ERR<n>STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR<n>STATUS.{V, UE, OF, CE, DE} are nonzero before the write.
- The write does not clear the nonzero ERR<n>STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR<n>STATUS are also defined as **UNKNOWN** where certain combinations of ERR<n>STATUS.{V, DE, UE} are zero. The rules for writes to ERR<n>STATUS allow a node to implement such a field as a fixed read-only value.

For example, when RAS System Architecture v1.1 is implemented, a write to ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as **UNKNOWN** values when ERR<n>STATUS.V is zero, this means those fields can be implemented as read-only if applicable.

To ensure correct and portable operation, when software is clearing the valid fields in the register to allow new errors to be recorded, Arm recommends that software performs the following sequence of operations in order:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- In a single write to ERR<n>STATUS:
  - Write ones to all the W1C fields that are nonzero in the read value.
  - Write zero to all the W1C fields that are zero in the read value.
  - Write zero to all the RW fields.

- Read back ERR<n>STATUS after the write to confirm no new fault has been recorded.

Otherwise, these fields might not have the correct value when a new fault is recorded.

Accessibility

ERR<n>STATUS.{AV, V, UE, ER, OF, MV, CE, DE, PN, UET, CI} are write-one-to-clear (W1C) fields, meaning writes of zero are ignored, and a write of one or all-ones to the field clears the field to zero. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is IMPLEMENTATION DEFINED. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, software must clear the valid fields in the register to allow new errors to be recorded. However, between reading the register and clearing the valid fields, a new error might have overwritten the register. To prevent this error being lost by software, the register prevents updates to fields that might have been updated by a new error.

When RAS System Architecture v1.0 is implemented:

- Writes to ERR<n>STATUS.{UE, DE, CE} are ignored if ERR<n>STATUS.OF is 1 and is not being cleared to 0.
- Writes to ERR<n>STATUS.V are ignored if any of ERR<n>STATUS.{UE, DE, CE} are nonzero and are not being cleared to zero.
- Writes to ERR<n>STATUS.{AV, MV} and the ERR<n>STATUS.{ER, PN, UET, IERR, SERR} syndrome fields are ignored if the highest priority nonzero error status field is not being cleared to zero. The error status fields in priority order from highest to lowest, are ERR<n>STATUS.UE, ERR<n>STATUS.DE, and ERR<n>STATUS.CE.

When RAS System Architecture v1.1 is implemented, a write to the register is ignored if all of:

- Any of ERR<n>STATUS.{V, UE, OF, CE, DE} are nonzero before the write.
- The write does not clear the nonzero ERR<n>STATUS.{V, UE, OF, CE, DE} fields to zero by writing ones to the applicable field or fields.

Some of the fields in ERR<n>STATUS are also defined as UNKNOWN where certain combinations of ERR<n>STATUS.{V, DE, UE} are zero. The rules for writes to ERR<n>STATUS allow a node to implement such a field as a fixed read-only value.

For example, when RAS System Architecture v1.1 is implemented, a write to ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as UNKNOWN values when ERR<n>STATUS.V is zero, this means those fields can be implemented as read-only if applicable.

To ensure correct and portable operation, when software is clearing the valid fields in the register to allow new errors to be recorded, Arm recommends that software performs the following sequence of operations in order:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- In a single write to ERR<n>STATUS:
  - Write ones to all the W1C fields that are nonzero in the read value.

- Write zero to all the W1C fields that are zero in the read value.
- Write zero to all the RW fields.

- Read back ERR<n>STATUS after the write to confirm no new fault has been recorded.

Otherwise, these fields might not have the correct value when a new fault is recorded.

Component	Offset	Instance	Range
RAS	0x10	ERR0STATUS	None

This interface is accessible as follows:

RW

B.8.4 ERR0ADDR, Error Record <n> Address Register

If an address is associated with a detected error, then it is written to ERR<n>ADDR when the error is recorded. It is **IMPLEMENTATION DEFINED** how the recorded address maps to the software-visible physical address. Software might have to reconstruct the actual physical addresses using the identity of the node and knowledge of the system.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record <n>. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record <n>. If the node owns a single record then FirstRecordOfNode(n) = n.

Attributes

Width

64

Component

RAS

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
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-224: ext_err0addr bit assignments

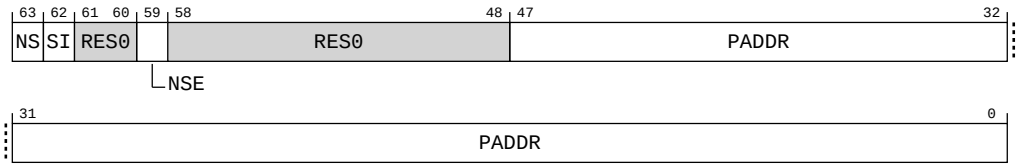


Table B-435: ERR0ADDR bit descriptions

Bits	Name	Description	Reset
[63]	NS	**When IsFeatureImplemented(FEAT_RME)** Non-secure attribute. With ERR<n>ADDR.NSE, indicates the physical address space of the recorded location. **0b0** When ERR<n>ADDR.NSE == 0: ERR<n>ADDR.PADDR is a Secure address. When ERR<n>ADDR.NSE == 1: ERR<n>ADDR.PADDR is a Root address. **0b1** When ERR<n>ADDR.NSE == 0: ERR<n>ADDR.PADDR is a Non-secure address. When ERR<n>ADDR.NSE == 1: ERR<n>ADDR.PADDR is a Realm address. **When !IsFeatureImplemented(FEAT_RME)** Non-secure attribute. **0b0** ERR<n>ADDR.PADDR is a Secure address. **0b1** ERR<n>ADDR.PADDR is a Non-secure address. **Otherwise** RES0	x

Bits	Name	Description	Reset
[62]	SI	**When IsFeatureImplemented(FEAT_RME)** Secure Incorrect. Indicates whether ERR<n>ADDR.{NS, NSE} are valid. **0b0** ERR<n>ADDR.{NS, NSE} are correct. That is, they match the programmers' view of the physical address space for the recorded location. **0b1** ERR<n>ADDR.{NS, NSE} might not be correct, and might not match the programmers' view of the physical address space for the recorded location. **When !IsFeatureImplemented(FEAT_RME)** Secure Incorrect. Indicates whether ERR<n>ADDR.NS is valid. **0b0** ERR<n>ADDR.NS is correct. That is, it matches the programmers' view of the Non-secure attribute for the recorded location. **0b1** ERR<n>ADDR.NS might not be correct, and might not match the programmers' view of the Non-secure attribute for the recorded location. **Otherwise** RES0	x
[61:60]	**RES0**	Reserved	**RES0**
[59]	NSE	**When IsFeatureImplemented(FEAT_RME)** Physical Address Space. Together with ERR<n>ADDR.NS, indicates the address space for ERR<n>ADDR.PADDR. **Otherwise** RES0	x
[58:48]	**RES0**	Reserved	**RES0**
[47:0]	PADDR	Physical Address [47:0]. Address of the recorded location	48{x}

Accessibility

Component	Offset	Instance	Range
RAS	0x18	ERR0ADDR	None

This interface is accessible as follows:

When ERRPFGF[FirstRecordOfNode].AV == '0' && ext-ERR0STATUS.AV == '1'

RO

Otherwise

RW

B.8.5 ERR0MISC0, Error Record <n> Miscellaneous Register 0

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record <n> implements a standard format Corrected error counter or counters (ERRFR[FirstRecordOfNode(n)].CEC != 0b000), then it is IMPLEMENTATION DEFINED whether error record <n> can record countable errors, and:

- If error record <n> records countable errors, then ERR<n>MISC0 implements the standard format Corrected error counter or counters for error record <n>.
- If error record <n> does not record countable errors, then it is recommended that the fields in ERR<n>MISC0 defined for the standard format counter or counters are **RES0**. That is, the fields behave like counters that never count.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record <n>. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record <n>. If the node owns a single record then FirstRecordOfNode(n) = n.

For IMPLEMENTATION DEFINED fields in ERR<n>MISC0, writing zero returns the error record to an initial quiescent state.

In particular, if any IMPLEMENTATION DEFINED syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Arm recommends that any IMPLEMENTATION DEFINED syndrome field that can generate a Fault Handling, Error Recovery, Critical, or IMPLEMENTATION DEFINED, interrupt request is disabled at Cold reset and is enabled by software writing an IMPLEMENTATION DEFINED nonzero value to an IMPLEMENTATION DEFINED field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Register offset

0x20

Access type

Read

R

Write

W

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-225: ext_err0misc0 bit assignments

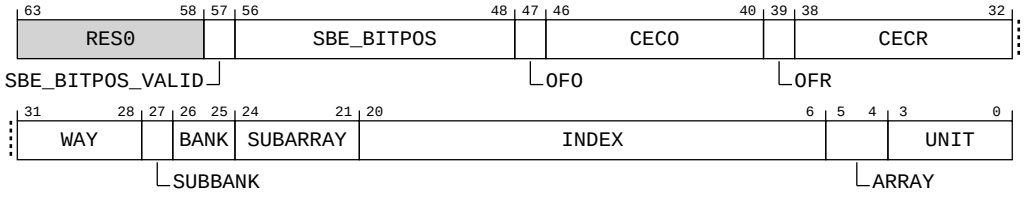


Table B-437: ERR0MISC0 bit descriptions

Bits	Name	Description	Reset
[63:58]	**RES0**	Reserved	**RES0**
[57]	SBE_BITPOS_VALID	Single Bit Error (SBE) bit position field ERXMISC0_EL1.SBE_BITPOS contains valid data **0b0** ERXMISC0_EL1.SBE_BITPOS does not contain valid data. **0b1** ERXMISC0_EL1.SBE_BITPOS contains valid data.	x
[56:48]	SBE_BITPOS	Single Bit Error (SBE) bit position. For a correctable error in a RAM with ECC (L1 data cache, L2 cache), indicates the bit position of the corrected error. Valid when ERXMISC0_EL1.SBE_BITPOS_VALID is 1'b1	9{x}

Bits	Name	Description	Reset
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERXMISC0_EL1.CECO is incremented and wraps through zero. **0b0** Other counter has not overflowed. **0b1** Other counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISC0_EL1.CECR. Unaffected by Cold or Warm reset.	7{x}
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERXMISC0_EL1.CECR is incremented and wraps through zero. **0b0** Repeat counter has not overflowed. **0b1** Repeat counter has overflowed. A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an **UNKNOWN** value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an **UNKNOWN** value. Unaffected by Cold or Warm reset.	x
[38:32]	CECR	Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome. This field resets to an IMPLEMENTATION DEFINED which might be **UNKNOWN** on a Cold reset. If the reset value is **UNKNOWN**, then the value of this field remains **UNKNOWN** until software initializes it. Unaffected by Cold or Warm reset.	7{x}

Bits	Name	Description	Reset
[31:28]	WAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Data Cache] • Indicates which Tag RAM way or data RAM way detected the error. Upper 2 bits are unused. [L2 TLB] • Indicates which RAM detected an error. The possible values are 0 (RAM 1) to 9 (RAM 10). [L1 Instruction Cache] • Indicates which way detected the error. Upper 2 bits are unused. [L2 Cache] • Indicates which way detected the error. Unaffected by Cold or Warm reset.	xxxx
[27]	SUBBANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L1 Instruction Cache] • Indicates which subbank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero. Unaffected by Cold or Warm reset.	x
[26:25]	BANK	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 bank detected the error. 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
[24:21]	SUBARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which L2 data 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

Bits	Name	Description	Reset
[20:6]	INDEX	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. [L1 Data Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size [L2 TLB] • Index of TLB RAM. Upper 4 bits are unused. [L1 Instruction Cache] • Indicates which index detected the error. Upper bits of the index are unused depending on the cache size. Unaffected by Cold or Warm reset.	`15{x}`

Bits	Name	Description	Reset
[5:4]	ARRAY	The encoding is dependent on the unit from which the error being recorded was detected. The possible values are: [L2 Cache] Indicates which array detected the error. The possible values are: • `0b00` L2 Tag RAM. • `0b01` L2 Data RAM. • `0b10` L2 TQ Data RAM. • `0b11` CHI Error. [L1 Data Cache] Indicates which array detected the error. The possible values are: • 00 LS Tag RAM 0. • 01 LS Tag RAM 1. • 10 LS Data RAM. • 11 LS Tag RAM 2. [L2 TLB] Indicates which array detected the error. The possible values are: • 00 Translation cache. • 01 GPT cache (when LEGACY_TZ_EN is 0). • 10 Reserved. • 11 Reserved. [L1 Instruction Cache] Indicates which array that detected the error, Data Array has higher priority. The possible values are: • `0b00` Tag. • `0b01` Data. Unaffected by Cold or Warm reset.	`xx`
[3:0]	UNIT	Indicates the unit which detected the error. The possible values are: **`0b0001`** L1 Instruction Cache. **`0b0010`** L2 TLB. **`0b0100`** L1 Data Cache. **`0b1000`** L2 Cache.	`xxxx`

Access

Reads from ERR<n>MISC0 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.


Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Accessibility

Reads from ERR<n>MISC0 return an IMPLEMENTATION DEFINED value and writes have IMPLEMENTATION DEFINED behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.


Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x20	ERR0MISC0	None

This interface is accessible as follows:

RW

B.8.6 ERR0MISC1, Error Record <n> Miscellaneous Register 1

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record <n>. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record <n>. If the node owns a single record then FirstRecordOfNode(n) = n.

For IMPLEMENTATION DEFINED fields in ERR<n>MISC1, writing zero returns the error record to an initial quiescent state.

In particular, if any IMPLEMENTATION DEFINED syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Arm recommends that any IMPLEMENTATION DEFINED syndrome field that can generate a Fault Handling, Error Recovery, Critical, or IMPLEMENTATION DEFINED, interrupt request is disabled at Cold reset and is enabled by software writing an IMPLEMENTATION DEFINED nonzero value to an IMPLEMENTATION DEFINED field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Register offset

0x28

Access type

Read

R

Write

W

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-226: ext_err0misc1 bit assignments

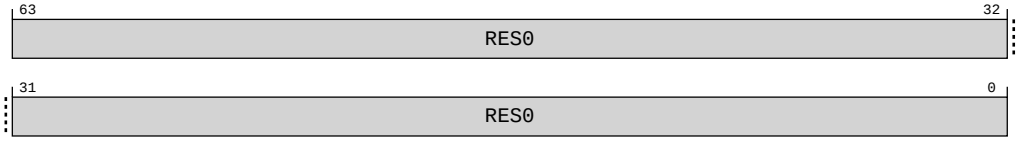


Table B-439: ERR0MISC1 bit descriptions

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

Access

Reads from ERR<n>MISC1 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.

Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Accessibility

Reads from ERR<n>MISC1 return an IMPLEMENTATION DEFINED value and writes have IMPLEMENTATION DEFINED behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.


Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x28	ERR0MISC1	None

This interface is accessible as follows:

RW

B.8.7 ERR0MISC2, Error Record <n> Miscellaneous Register 2

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record <n>. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record <n>. If the node owns a single record then FirstRecordOfNode(n) = n.

For IMPLEMENTATION DEFINED fields in ERR<n>MISC2, writing zero returns the error record to an initial quiescent state.

In particular, if any IMPLEMENTATION DEFINED syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Arm recommends that if RAS System Architecture v1.1 is not implemented then ERR<n>MISC2 does not require zeroing to return the record to a quiescent state.


Note

Arm recommends that any IMPLEMENTATION DEFINED syndrome field that can generate a Fault Handling, Error Recovery, Critical, or IMPLEMENTATION DEFINED, interrupt request is disabled at Cold reset and is enabled by software writing an IMPLEMENTATION DEFINED nonzero value to an IMPLEMENTATION DEFINED field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Register offset

0x30

Access type

Read

R

Write

W

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```


Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-227: ext_err0misc2 bit assignments

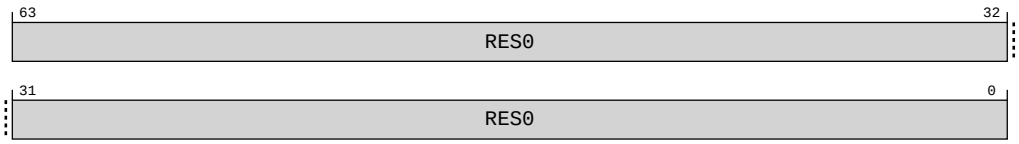


Table B-441: ERR0MISC2 bit descriptions

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

Access

Reads from ERR<n>MISC2 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Accessibility

Reads from ERR<n>MISC2 return an IMPLEMENTATION DEFINED value and writes have IMPLEMENTATION DEFINED behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.

- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.


Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x30	ERR0MISC2	None

This interface is accessible as follows:

RW

B.8.8 ERR0MISC3, Error Record <n> Miscellaneous Register 3

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to locate where the error was detected.
- If the error was detected within a FRU, the identity of the FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record n supports the RAS Timestamp Extension (ERRFR[FirstRecordOfNode(n)].TS != 0b00), then ERR<n>MISC3 contains the timestamp value for error record n when the error was detected. Otherwise the contents of ERR<n>MISC3 are IMPLEMENTATION DEFINED.

Configurations

ERRFR[FirstRecordOfNode(n)] describes the features implemented by the node that owns error record <n>. FirstRecordOfNode(n) is the index of the first error record owned by the same node as error record <n>. If the node owns a single record then FirstRecordOfNode(n) = n.

For IMPLEMENTATION DEFINED fields in ERR<n>MISC3, writing zero returns the error record to an initial quiescent state.

In particular, if any IMPLEMENTATION DEFINED syndrome fields might generate a Fault Handling or Error Recovery Interrupt request, writing zero is sufficient to deactivate the Interrupt request.

Fields that are read-only, nonzero, and ignore writes are compliant with this requirement.

Arm recommends that if RAS System Architecture v1.1 is not implemented then ERR<n>MISC3 does not require zeroing to return the record to a quiescent state.

Note

Arm recommends that any IMPLEMENTATION DEFINED syndrome field that can generate a Fault Handling, Error Recovery, Critical, or IMPLEMENTATION DEFINED, interrupt request is disabled at Cold reset and is enabled by software writing an IMPLEMENTATION DEFINED nonzero value to an IMPLEMENTATION DEFINED field in ERRCTLR[FirstRecordOfNode(n)].

Attributes

Width

64

Component

RAS

Register offset

0x38

Access type

Read

R

Write

W

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-228: ext_err0misc3 bit assignments

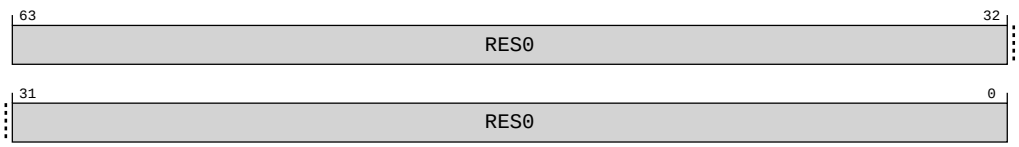


Table B-443: ERR0MISC3 bit descriptions

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

Access

Reads from ERR<n>MISC3 return an **IMPLEMENTATION DEFINED** value and writes have **IMPLEMENTATION DEFINED** behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.


Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Accessibility

Reads from ERR<n>MISC3 return an IMPLEMENTATION DEFINED value and writes have IMPLEMENTATION DEFINED behavior.

If the Common Fault Injection Mechanism is implemented by the node that owns this error record, and ERRPFGF[FirstRecordOfNode(n)].MV is 1, then some parts of this register are read/write when ext-ERR<n>STATUS.MV is 0. See ext-ERR<n>PFGF.MV for more information.

For other parts of this register, or if the Common Fault Injection Mechanism is not implemented, then Arm recommends that:

- Miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.
- When ext-ERR<n>STATUS.MV is 1, the miscellaneous syndrome specific to the most recently recorded error ignores writes.


Note

These recommendations allow a counter to be reset in the presence of a persistent error, while preventing specific information, such as that identifying a FRU, from being lost if an error is detected while the previous error is being logged.

Component	Offset	Instance	Range
RAS	0x38	ERR0MISC3	None

This interface is accessible as follows:

RW

B.8.9 ERR0PFGF, Error Record <n> Pseudo-fault Generation Feature Register

Defines which common architecturally-defined fault generation features are implemented.

Configurations

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x800

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x111 xxxx xxxx xxxx xxx1 1010 0110 0011
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-229: ext_err0pfgf bit assignments

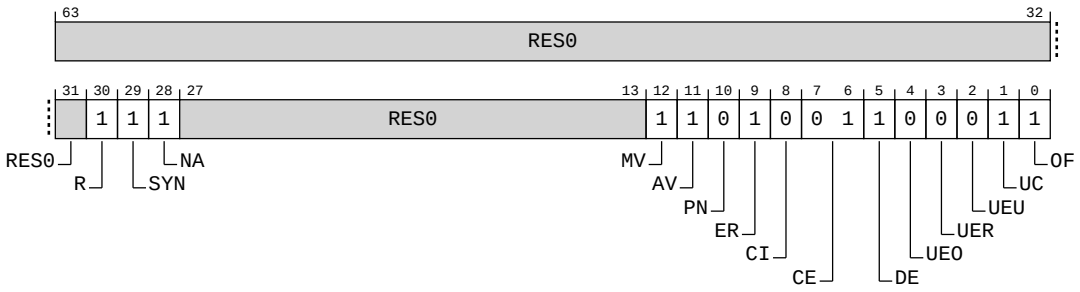


Table B-445: ERR0PFGF bit descriptions

Bits	Name	Description	Reset
[63:31]	**RES0**	Reserved	**RES0**
[30]	R	Restartable. Support for Error Generation Counter restart mode. **0b1** Error Generation Counter restart mode is implemented and is controlled by ext-ERR<n>PFGCTL.R.	0b1
[29]	SYN	Syndrome. Fault syndrome injection. **0b1** When an injected error is recorded, the node does not update the ext-ERR<n>STATUS.{IERR, SERR} fields. ext-ERR<n>STATUS.{IERR, SERR} are writable when ext-ERR<n>STATUS.V is 0.	0b1
[28]	NA	No access required. Defines whether this component fakes detection of the error on an access to the component or spontaneously in the fault injection state. **0b1** The component fakes detection of the error spontaneously in the fault injection state.	0b1
[27:13]	**RES0**	Reserved	**RES0**
[12]	MV	Miscellaneous syndrome. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. **0b1** When an injected error is recorded, the node might update some, but not all ERR<n>MISC<m> syndrome fields: • If any syndrome is recorded by the node in the ERR<n>MISC<m> registers, then ext-ERR<n>STATUS.MV is set to 1. • Otherwise, ext-ERR<n>STATUS.MV is set to ext-ERR<n>PFGCTL.MV.	0b1
[11]	AV	Address syndrome. Defines whether software can control the address recorded in ext-ERR<n>ADDR when an injected error is recorded. **0b1** When an injected error is recorded, the node does not update ext-ERR<n>ADDR and does: • Sets ext-ERR<n>STATUS.AV to 1.	0b1
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.PN status flag. **0b0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.PN to 0.	0b0
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.ER status flag. **0b1** When an injected error is recorded, ext-ERR<n>STATUS.ER is set to ext-ERR<n>PFGCTL.ER. This behavior replaces the architecture-defined rules for setting the ER bit.	0b1
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.CI status flag. **0b0** The node does not support this type of flag. This behavior replaces the architecture-defined rules for setting the CI bit.	0b0

Bits	Name	Description	Reset
[7:6]	CE	Corrected Error generation. Describes the types of Corrected error that the fault generation feature of the node can generate. **0b01** The fault generation feature of the node allows generation of a non-specific Corrected error, that is, a Corrected error that is recorded by setting ext-ERR<n>STATUS.CE to 0b10.	0b01
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate Deferred errors. **0b1** The fault generation feature of the node allows generation of Deferred errors.	0b1
[4]	UEO	Latent or Restartable Error generation. Describes whether the fault generation feature of the node can generate Latent or Restartable errors. **0b0** The fault generation feature of the node does not generate Latent or Restartable errors.	0b0
[3]	UER	Signaled or Recoverable Error generation. Describes whether the fault generation feature of the node can generate Signaled or Recoverable errors. **0b0** The fault generation feature of the node does not generate Signaled or Recoverable errors.	0b0
[2]	UEU	Unrecoverable Error generation. Describes whether the fault generation feature of the node can generate Unrecoverable errors. **0b0** The fault generation feature of the node does not generate Unrecoverable errors.	0b0
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate Uncontainable errors. **0b1** The fault generation feature of the node allows generation of Uncontainable errors.	0b1
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.OF status flag. **0b1** When an injected error is recorded, ext-ERR<n>STATUS.OF is set to ext-ERR<n>PFGCTL.OF. This behavior replaces the architecture-defined rules for setting the OF bit.	0b1

Accessibility

Component	Offset	Instance	Range
RAS	0x800	ERR0PFGF	None

This interface is accessible as follows:

RO

B.8.10 ERR0PFGCTL, Error Record <n> Pseudo-fault Generation Control Register

Enables controlled fault generation.

Configurations

ext-ERR<n>PFGF describes the Common Fault Injection features implemented by the node.

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x808

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 00xx xxxx xxxx xxxx xxxx x0x0 000x xx00
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3 0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-230: ext_err0pfgctl bit assignments

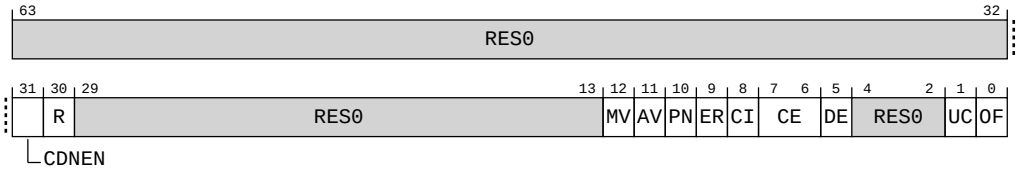


Table B-447: ERR0PFGCTL bit descriptions

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]	MV	Miscellaneous syndrome. The value written to AArch64-ERXSTATUS.MV when an injected error is recorded. **0b00** AArch64-ERXSTATUS.MV is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.MV is set to 1 when an injected error is recorded.	x
[11]	AV	Address syndrome. The value written to AArch64-ERXSTATUS.AV when an injected error is recorded. **0b00** AArch64-ERXSTATUS.AV is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.AV is set to 1 when an injected error is recorded.	x
[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]	ER	Error Reported flag. The value written to AArch64-ERXSTATUS.ER when an injected error is recorded. **0b00** AArch64-ERXSTATUS.ER is set to 0 when an injected error is recorded. **0b01** AArch64-ERXSTATUS.ER is set to 1 when an injected error is recorded.	x
[8]	CI	Critical Error flag. The value that is written to AArch64-ERXSTATUS.CI when an injected error is recorded. **0b0** AArch64-ERXSTATUS.CI is set to 0 when an injected error is recorded. **0b1** AArch64-ERXSTATUS.CI is set to 1 when an injected error is recorded.	0b0

Bits	Name	Description	Reset
[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are: **0b00** No error of this type will be generated. **0b01** A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10, might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b00
[5]	DE	Deferred Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[4:2]	**RES0**	Reserved	**RES0**
[1]	UC	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0
[0]	OF	Uncontainable Error generation enable. The possible values are: **0b0** No error of this type will be generated. **0b1** An error of this type might be generated when the Error Generation Counter decrements to zero. Cold reset only. Unaffected by Warm reset	0b0

Accessibility

Component	Offset	Instance	Range
RAS	0x808	ERR0PFGCTL	None

This interface is accessible as follows:

RW

B.8.11 ERR0PFGCDN, Error Record <n> Pseudo-fault Generation Countdown Register

Generates one of the errors enabled in the corresponding ext-ERR<n>PFGCTL register.

Configurations

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x810

Access type

RW

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |
63   59   55   51   47   43   39   35   31   27   23   19   15   11   7    3  0
```

Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-231: ext_err0pfgcdn bit assignments

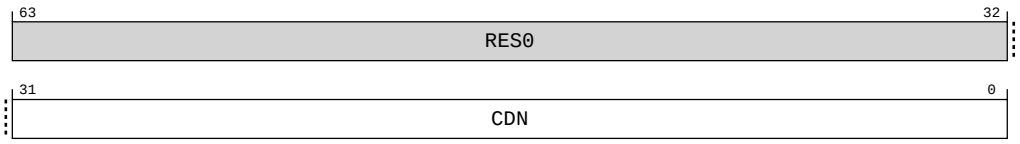


Table B-449: ERR0PFGCDN bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:0]	CDN	Countdown value. This field is copied to Error Generation Counter when either: • Software writes ERXPFGCTL_EL1.CDNEN with 1. • The Error Generation Counter decrements to zero and ERXPFGCTL_EL1.R == 0b1. Unaffected by Cold or Warm reset. **Note:** The current Error Generation Counter value is not visible to software.	32{x}

Accessibility

Component	Offset	Instance	Range
RAS	0x810	ERR0PFGCDN	None

This interface is accessible as follows:

RW

Appendix C Document revisions

This appendix records the changes between released issues of this document.

C.1 Revisions

Changes between released issues of this book are summarized in tables.

The first table is for the first release. Then, each table compares the new issue of the book with the last released issue of the book. Release numbers match the revision history in Release Information on page 2.

Table C-1: Issue 0000-02

Change	Location
First early access release for r0p0	
Updated section	2.1 Neoverse V3 core features on page 27
Added details on warm reset and reworded paragraph	4. Clocks and resets on page 46
Added details on responses from MMU and external aborts	6.6 Responses on page 63
Updated for CHI.F	9. L2 memory system on page 76
Updated section	11. RAS Extension support on page 97
Added bits describing FEAT_ECBHB	A.6.15 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1 on page 441
Added bits describing RFPEN and TFPEN	A.12.4 ERXCTLR_EL1, Selected Error Record Control Register on page 655
Added bit description for IERR	A.12.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register on page 659
Updated bit descriptions	A.12.10 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0 on page 682
Updated translation regime information	6.3 Translation Lookaside Buffer match process on page 61
Updated number of system registers and updated table	10. Direct access to internal memory on page 78
Updated table and added information about 3MB L2 cache	10.2 L2 cache encodings on page 87
Updated values in tables	10.2.1 L2 tag RAM returned data on page 89
Added table for data register 2	10.2.2 L2 data RAM returned data on page 91
Updated data for tables in section	10.2.3 L2 TLB RAM returned data on page 92
Updated bit field values	10.2.4 L2 Victim RAM returned data on page 95
Updated values for CTI identification registers	18.7 CTI register identification values on page 125
Updated bit field values	A.4.1 IMP_CPURNDBR_EL3, CPU Random Number Base Register on page 400
Updated product name	A.6.1 MIDR_EL1, Main ID Register on page 406
Updated bit field description and reset value	A.6.18 MPAMIDR_EL1, MPAM ID Register (EL1) on page 449
Updated bit field descriptions	A.8.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0 on page 473

Change	Location
Updated bit field description	A.12.2 ERRSELR_EL1, Error Record Select Register on page 649
Updated bit field descriptions	A.12.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register on page 669
Added reset values	B.4.25 EDDFR, External Debug Feature Register on page 1257
Updated bit field description	B.7.20 TRCDEVARCH, Device Architecture Register on page 1357
Updated details on Shareability for Normal memory	6.7 Memory behavior and supported memory types on page 65
Reworded paragraph in section	8.5 Data prefetching on page 75
Updated PMU events	19.1 Performance monitors events on page 130
Removed irrelevant information	7.5 Instruction cache hardware coherency on page 71
Removed reference to L3 cache	8.1 L1 data cache behavior on page 72
Editorial update	9.2 Support for memory types on page 76
Removed reference to L3 cache	5.2.1 Wait for Interrupt and Wait for Event on page 48
Removed reference to L3 cache	6.4 Translation table walks on page 62
Removed irrelevant information	8.3 Instruction implementation in the L1 data memory system on page 74
Added information about L2 cache	9.1 L2 cache on page 76
Added chapter	17. Random number generator support on page 117
Removed references to L3 cache	A.6.20 CLIDR_EL1, Cache Level ID Register on page 454

Table C-2: Differences between Issue 0000-02 and Issue 0001-03

Change	Location
First early access release for r0p1	-
Editorial revisions	Throughout document
Added product revision information	2.7 Product revisions on page 39
Updated information in registers	A. AArch64 registers on page 183
Updated information in registers	B. External registers on page 1060
Updated section	4. Clocks and resets on page 46
Updated bit field description	A.1.10 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 208
Updated value in table	18.6 CoreSight component identification on page 124
Updated section	5.1 Voltage and power domains on page 47

Table C-3: Differences between Issue 0001-03 and Issue 0001-04

Change	Location
Second early access release for r0p1	-
Editorial revisions	Throughout document
Updated product name	Throughout document
Updated PMU counter events: PMU description for event 0x0050 L2D_CACHE_RD	19.1 Performance monitors events on page 130
Updated fields description and range	10.1.3 L1 instruction TLB returned data on page 81
Updated fields description and range	10.1.6 L1 data TLB returned data on page 85
Updated fields description and range	10.2 L2 cache encodings on page 87

Change	Location
• Updated tables title for tables 'L2 TLB format for Instruction Register 0 if bit[6] is set to 0' and 'L2 TLB format for Instruction Register 0 if bit[6] is set to 1'. • Updated fields description in table 'L2 TLB format for Instruction Register 2'	10.2.3 L2 TLB RAM returned data on page 92
Updated information in registers	A. AArch64 registers on page 183
Updated information in registers	B. External registers on page 1060