



Arm[®] Cortex[®]-R82 Processor

Revision r3p1

Technical Reference Manual

Non-Confidential

Issue 05

Copyright © 2022–2024 Arm Limited (or its affiliates). 102670_0301_05_en
All rights reserved.



Arm® Cortex®-R82 Processor Technical Reference Manual

This document is Non-Confidential.

Copyright © 2022–2024 Arm Limited (or its affiliates). All rights reserved.

This document is protected by copyright and other intellectual property rights.

Arm only permits use of this document if you have reviewed and accepted [Arm's Proprietary Notice](#) found at the end of this document.

This document (102670_0301_05_en) was issued on 2024-10-25. There might be a later issue at <https://developer.arm.com/documentation/102670>

The product revision is r3p1.

See also: [Proprietary Notice](#) | [Product and document information](#) | [Useful resources](#)

Start Reading

If you prefer, you can skip to [the start of the content](#).

Intended audience

This manual is for system designers, system integrators, and programmers who are designing or programming a *System on Chip* (SoC) that uses the Cortex®-R82 processor.

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.

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>.

Contents

1. The Cortex®-R82 processor.....	12
1.1 About the Cortex®-R82 processor.....	12
1.2 Features.....	13
1.3 Configuration options.....	15
1.3.1 Configuration parameters.....	15
1.3.2 Integration-time configuration options.....	21
1.4 Supported standards and specifications.....	23
1.5 Design process.....	25
1.6 Documentation.....	26
1.7 Product revisions.....	27
2. Technical overview.....	28
2.1 Terminology.....	28
2.2 Components.....	28
2.3 Interfaces.....	35
3. Programmers' model.....	39
3.1 About the programmers' model.....	39
3.2 Arm®v8-R AArch64 architecture concepts.....	39
3.2.1 Architecture requirements.....	40
3.2.2 Execution state.....	44
3.2.3 Exception levels.....	44
3.2.4 Instruction set architecture.....	46
3.2.5 Data types.....	46
3.2.6 Arm®v8-R AArch64 registers.....	46
3.2.7 Memory model.....	48
3.2.8 Security model.....	49
3.3 Advanced SIMD and floating-point.....	50
4. Clocks and resets.....	51
4.1 Clocks and clock enables.....	51
4.2 Clock domains.....	52
4.3 Resets.....	54

4.4 Resetting with PPU.....	55
5. Power management.....	56
5.1 About power management.....	56
5.2 Voltage domain.....	56
5.3 Clock gating.....	56
5.3.1 Local and regional clock gating.....	57
5.3.2 Hierarchical clock gating.....	57
5.4 Power domains.....	59
5.5 Power mode control.....	61
5.6 Debug over powerdown.....	62
5.7 Core power modes and transitions.....	62
5.8 Core powerdown.....	67
5.9 Cluster power modes.....	68
5.10 Cluster operating modes.....	72
5.11 Cluster PPU mode transitions.....	73
5.11.1 Rules governing cluster PPU mode transitions.....	76
5.11.2 PPU mode transition behavior.....	76
5.11.3 DebugBlock power modes.....	78
5.12 Cluster powerdown.....	78
5.13 Cluster power mode and core power mode dependencies.....	78
6. Power and reset control with PPUs.....	80
6.1 The Power Policy Unit.....	80
6.2 PPU operation.....	83
6.2.1 Implicit resets from PPU mode change.....	83
6.3 Utility bus accesses.....	84
6.4 Encodings for cluster power modes and operating modes.....	84
6.5 Encodings for core power modes.....	85
6.6 Programming sequences for the cluster and the core.....	86
6.6.1 Programming sequence to bring the cluster and cores from Off to On mode.....	86
6.6.2 Programming sequence to bring the cluster and cores from On to Off mode.....	87
6.6.3 Programming sequence for an interrupt controller to bring cores and cluster to On or Off mode.....	88
6.7 Explicit resetting of cluster and cores and debug recovery.....	89
6.8 ECC errors during power transitions.....	93
6.9 Implications of not having a System Control Processor.....	94

6.10 PPU and reset management register summary.....	94
7. Initialization.....	95
7.1 Initializing the Cortex®-R82 processor.....	95
7.2 Initializing TCMs.....	97
7.2.1 Preloading TCMs.....	98
7.2.2 Preloading TCMs with ECC.....	98
7.2.3 Using TCMs from reset.....	99
7.3 Initializing LLRAM.....	100
7.3.1 Preloading LLRAM.....	100
7.3.2 Using LLRAM from reset.....	101
7.4 Disabling EL2.....	102
8. Memory system.....	103
8.1 About the memory system.....	103
8.2 TCM memories.....	106
8.3 L1 memory system.....	108
8.3.1 L1 instruction memory system.....	110
8.3.2 L1 data memory system.....	113
8.4 L2 memory system.....	120
8.4.1 L2 cache slice integration.....	121
8.4.2 L2 cache allocation policy.....	123
8.4.3 L2 cache partitioning.....	123
8.4.4 L2 cache stashing.....	125
8.4.5 L2 cache data RAM latency.....	125
8.5 LLPP manager interface.....	126
8.5.1 LLPP features.....	128
8.5.2 LLPP memory attributes.....	128
8.5.3 LLPP transfers.....	128
8.5.4 LLPP AXI transfer restrictions.....	129
8.6 SPP manager interface.....	132
8.6.1 SPP features.....	134
8.6.2 SPP memory attributes.....	135
8.6.3 SPP transfers.....	135
8.6.4 SPP AXI transfer restrictions.....	136
8.7 MM interface.....	140
8.7.1 CHI requester interface.....	141

8.7.2 AXI manager interface.....	149
8.8 LLRAM manager interface.....	155
8.8.1 LLRAM features.....	158
8.8.2 LLRAM attributes.....	158
8.8.3 LLRAM transactions.....	160
8.8.4 Support for memory types.....	161
8.8.5 LLRAM write response.....	162
8.8.6 AXI4 compatibility mode.....	162
8.8.7 LLRAM privilege information.....	163
8.9 MACP subordinate interface.....	163
8.9.1 MACP features.....	164
8.9.2 MACP attributes.....	164
8.9.3 MACP transaction types.....	165
8.10 ACELS interface.....	167
8.10.1 ACELS features.....	169
8.10.2 ACELS attributes.....	170
8.10.3 ACELS transaction types.....	171
8.10.4 TCM subordinate.....	172
8.10.5 LLRAM ACP.....	175
8.11 Utility bus.....	176
8.11.1 Utility bus accesses.....	177
8.11.2 Base addresses for system components.....	178
8.12 Direct access to internal memories.....	178
8.12.1 Direct access to L1 memory.....	179
8.12.2 Direct access to L2 and LCU memory.....	179
8.13 Exclusives and atomics support.....	180
8.14 Real-time considerations.....	182
8.14.1 Interrupt latency.....	183
8.14.2 Real-time hierarchy.....	187
8.14.3 Freedom from Interference.....	188
8.14.4 Quality of Service.....	193
9. Memory management.....	195
9.1 About the memory management.....	195
9.2 MPU.....	196
9.2.1 MPU regions.....	197

9.2.2 Virtualization support.....	206
9.2.3 MPU register access.....	208
9.3 MMU.....	209
9.3.1 TLB organization.....	210
9.3.2 TLB match process.....	212
9.3.3 Translation table walks.....	212
9.3.4 MMU memory accesses.....	213
9.3.5 Responses.....	214
9.3.6 Memory behavior and supported memory types.....	216
9.3.7 Page-based hardware attributes.....	217
10. RAS Extension support.....	218
10.1 RAS Extension support in the processor.....	218
10.2 Memory protection behavior.....	219
10.3 Error containment.....	223
10.4 Fault detection and reporting.....	225
10.5 Error detection and reporting.....	226
10.5.1 Error reporting and performance monitoring.....	228
10.6 Error injection.....	228
10.7 RAS register summary.....	229
11. GIC CPU interface.....	230
11.1 About the GIC CPU interface.....	230
11.2 Disabling the GIC CPU interface.....	232
11.3 Bypassing the GIC CPU interface.....	232
11.4 GIC CPU interface register summary.....	233
12. Generic Timer.....	234
12.1 About the Generic Timer.....	234
12.2 Generic Timer functional description.....	234
12.3 Generic Timer register summary.....	234
13. Debug.....	235
13.1 About debug methods.....	235
13.2 Debug functional description.....	237
13.3 Debug register accesses.....	239
13.3.1 Processor accesses.....	239

13.3.2 Effects of resets on Debug registers.....	240
13.3.3 External access permissions to debug registers.....	240
13.3.4 Breakpoints and watchpoints.....	241
13.4 Debug events.....	241
13.4.1 Watchpoint debug events.....	242
13.4.2 Debug OS Lock.....	242
13.5 The DebugBlock.....	242
13.5.1 DebugBlock components.....	244
13.6 ECT.....	245
13.6.1 Supported debug and trace trigger events.....	246
13.6.2 CTI triggers.....	247
13.6.3 CTI register summary.....	248
13.7 Debug register summary.....	248
14. PMU.....	249
14.1 About the PMU.....	249
14.2 PMU functional description.....	250
14.3 External register access permissions to the PMU registers.....	251
14.4 PMU events.....	251
14.4.1 Core PMU events.....	251
14.4.2 Cluster PMU events.....	262
14.5 PMU interrupts.....	266
14.6 PMU register summary.....	266
15. ETM.....	267
15.1 About the ETM.....	267
15.2 ETM trace unit generation options and resources.....	268
15.3 ETM event connectivity.....	270
15.4 Operation.....	270
15.4.1 Precise TraceEnable events.....	270
15.4.2 Parallel instruction execution.....	271
15.4.3 Comparator features.....	271
15.4.4 Trace features.....	271
15.4.5 Packet formats.....	271
15.4.6 Resource selection.....	272
15.4.7 Trace flush behavior.....	273
15.4.8 Low-power state behavior.....	273

15.4.9 Cycle counter.....	274
15.4.10 Non-architectural exceptions.....	274
15.4.11 Trace synchronization.....	274
15.5 Modes of operation and execution.....	274
15.5.1 Use of the ETM main enable bit.....	274
15.5.2 Programming and reading ETM registers.....	276
15.5.3 External register access permissions.....	276
15.6 ETM register summary.....	276
16. Advanced SIMD and floating-point support.....	277
16.1 About the Advanced SIMD and floating-point support.....	277
16.2 Low-latency single-precision instructions.....	277
A. AArch64 registers.....	279
A.1 AArch64 register summaries.....	279
A.1.1 AArch64 Identification registers summary.....	279
A.1.2 AArch64 Generic System Control registers summary.....	280
A.1.3 AArch64 Performance Monitors registers summary.....	285
A.1.4 AArch64 System instructions summary.....	287
A.1.5 AArch64 Debug registers summary.....	294
A.1.6 AArch64 GIC system registers summary.....	295
A.1.7 AArch64 RAS registers summary.....	297
A.1.8 AArch64 Trace unit registers summary.....	298
A.1.9 AArch64 Special-purpose registers summary.....	301
A.1.10 AArch64 Generic Timer registers summary.....	301
A.1.11 AArch64 Other system control registers summary.....	302
A.2 AArch64 register descriptions.....	303
A.2.1 AArch64 Identification register description.....	303
A.2.2 AArch64 Generic System control register description.....	388
A.2.3 AArch64 Performance Monitors register description.....	710
A.2.4 AArch64 System instruction register description.....	815
A.2.5 AArch64 Debug register description.....	1025
A.2.6 AArch64 GIC register description.....	1085
A.2.7 AArch64 RAS register description.....	1184
A.2.8 AArch64 Trace register description.....	1216
A.2.9 AArch64 Special purpose register description.....	1353
A.2.10 AArch64 Generic Timer register description.....	1376

A.2.11 AArch64 other register description.....	1407
B. External registers.....	1415
B.1 Registers accessed over the Utility bus.....	1415
B.1.1 Register summaries for registers accessed over the Utility bus.....	1416
B.1.2 Register descriptions for registers accessed over the Utility bus.....	1420
B.2 Registers accessed over the Debug APB bus.....	1598
B.2.1 Register summaries for registers accessed over the Debug APB bus.....	1601
B.2.2 Register descriptions for registers accessed over the Debug APB bus.....	1615
C. Processor UNPREDICTABLE behaviors.....	2177
C.1 SBZ or SBO fields in instructions.....	2177
C.2 CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values.....	2178
C.3 CONSTRAINED UNPREDICTABLE behavior due to inadequate context synchronization....	2179
C.4 Translation table base address alignment.....	2179
C.5 The Performance Monitors Extension.....	2180
C.5.1 CONSTRAINED UNPREDICTABLE accesses to PMXEVTYPER_ELO or PMXEVTYPER_ELO.....	2180
C.5.2 CONSTRAINED UNPREDICTABLE accesses to PMEVCNTR<n>_ELO and PMEVTYPER<n>_ELO.....	2181
C.5.3 CONSTRAINED UNPREDICTABLE behavior caused by MDCR_EL2.HPMN.....	2181
C.6 The Activity Monitors Extension.....	2181
C.7 Syndrome register handling for CONSTRAINED UNPREDICTABLE instructions treated as UNDEFINED.....	2182
C.8 Out of range virtual address.....	2182
C.9 Mapping of non-idempotent memory locations using the Normal memory type.....	2182
C.10 Instruction fetches from Device memory.....	2183
C.11 Programming the CSSELR_EL1.Level for a cache level that is not implemented.....	2183
C.12 Crossing a page boundary with different memory types or Shareability attributes.....	2184
C.13 CONSTRAINED UNPREDICTABLE behaviors with Load-Exclusive/Store-Exclusive pairs...	2184
C.14 CONSTRAINED UNPREDICTABLE behavior for instructions.....	2185
C.14.1 LDAXP, LDNP, LDNP (SIMD&FP).....	2185
C.14.2 LDP.....	2186
C.14.3 LDP (SIMD&FP).....	2186
C.14.4 LDPSW.....	2186
C.14.5 LDR (immediate), LDRB (immediate), LDRH (immediate), LDRSB (immediate), LDRSH (immediate), LDRSW (immediate).....	2187
C.14.6 LDXP.....	2187

C.14.7 STP.....	2187
C.14.8 STLXP.....	2188
C.14.9 STLXR, STLXRB, STLXRH.....	2188
C.14.10 STR (immediate), STRB (immediate), STRH (immediate).....	2188
C.14.11 STXP.....	2189
C.14.12 STXR, STXRB, STXRH.....	2189
C.15 Out of range values of the Set/Way/Index fields in cache maintenance instructions.....	2189
C.16 Reserved values in System and memory-mapped registers and translation table entries....	2190
C.17 CONSTRAINED UNPREDICTABLE behavior in Debug state.....	2190
C.17.1 Instructions that are CONSTRAINED UNPREDICTABLE in Debug state.....	2190
C.17.2 Exiting Debug state.....	2192
C.17.3 Changing the value of EDECR.SS when not in Debug state.....	2192
C.17.4 Syndrome information on Halting Step.....	2192
C.17.5 Illegal Execution state exception.....	2193
C.17.6 Alignment constraints.....	2193
C.17.7 Cumulative error flag.....	2193
C.17.8 Restart request trigger event.....	2194
C.17.9 External debug interface accesses to registers in reset.....	2194
C.17.10 Reserved and unallocated registers.....	2194
C.17.11 External accesses to DBGBVR<n>_EL1 and DBGBCR<n>_EL1.....	2195
C.17.12 External accesses to DBGWVR<n>_EL1 and DBGWCR<n>_EL1.....	2195
C.17.13 Accessing the EDESR.....	2195
C.17.14 Accessing the CTIAPPPULSE.....	2195
C.18 RAS registers.....	2196
D. Generic handler example.....	2197
D.1 Generic handler example, part 1.....	2197
D.2 Generic handler example, part 2.....	2198
Proprietary Notice.....	2199
Product and document information.....	2201
Product status.....	2201
Revision history.....	2201
Conventions.....	2203
Useful resources.....	2206

1. The Cortex®-R82 processor

This chapter provides an overview of the Cortex®-R82 processor and its features.

1.1 About the Cortex®-R82 processor

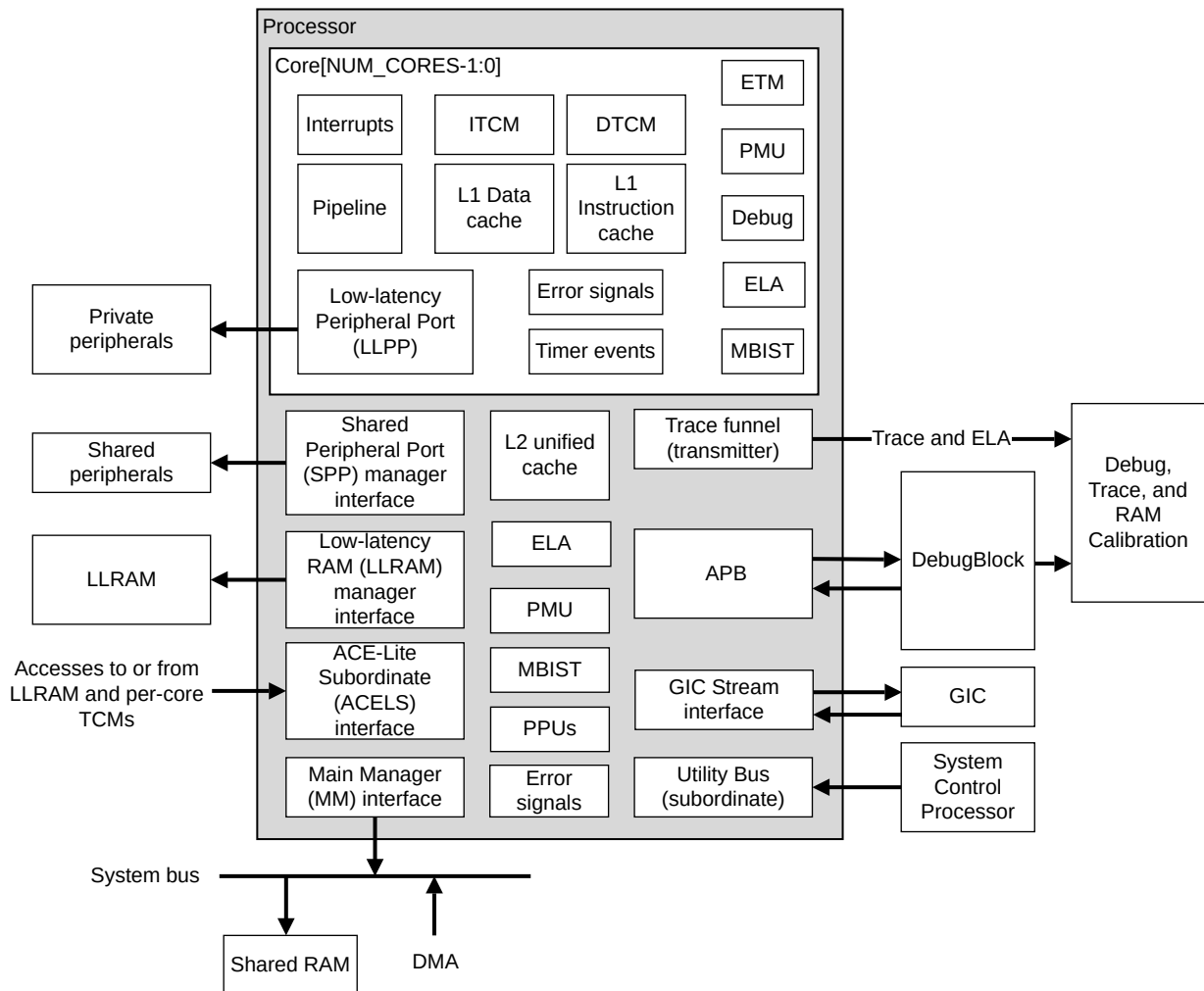
The Cortex®-R82 processor is a mid-performance, multi-core, in-order, superscalar processor for use in real-time embedded applications. The Cortex®-R82 processor implements the Arm®v8-R AArch64 architecture.

The Arm®v8-R AArch64 architecture is a 64-bit R-profile Arm® architecture with only AArch64 Execution state. Therefore, the Cortex®-R82 processor does not support the AArch32 Execution state. The Cortex®-R82 processor supports all the mandatory features from the Arm®v8.4 architecture as well as the prefetch speculation protection, the debug over powerdown, the *Reliability, Availability, and Serviceability* (RAS) Extension, and the Performance Monitors Extension.

The Cortex®-R82 processor is targeted at mobile modem and storage controller applications that require minimized latencies. The Cortex®-R82 processor has one to eight cores, each implementing a single Arm®v8-R AArch64 compliant *Processing Element* (PE). In the context of the Cortex®-R82 processor, the PE and core are conceptually the same.

The Cortex®-R82 processor supports multiple *Protected Memory System Architecture* (PMSA) and *Virtual Memory System Architecture* (VMSA) contexts to execute on the same core and uses virtualization technology to isolate them in memory space. Similarly, the Cortex®-R82 processor enables the real-time performance of different contexts to be isolated in time, which limits the impact of one context on the response time and determinism of a more critical context.

The following figure shows an example Cortex®-R82 processor system.

Figure 1-1: Example processor system

1.2 Features

The Cortex®-R82 processor has the following features:

Processor features

- 64-bit capability based on the Arm®v8-R AArch64 architecture.
- Up to eight cores with in-cluster hardware coherency.
- Optional inter-cluster hardware coherency with AMBA® CHI.
- A64 instruction set for the Arm®v8-R AArch64 architecture.
- AArch64 Execution state at EL0, EL1, and EL2 Exception levels (without EL3).
- Eight-stage, in-order superscalar pipeline with direct and indirect branch prediction.

- Support for *Protected Memory System Architecture* (PMSA) at EL1 and EL2 and optional support for *Virtual Memory System Architecture* (VMSA) at EL1.
- Secure state operation only at all Exception levels (Non-secure operation not supported).
- Compatible with Arm® TrustZone® technology, that is, support for accesses to both Secure and Non-secure physical memory address space.
- Optional *Advanced Single Instruction Multiple Data* (SIMD) and floating-point architecture support with two 64-bit data engine pipelines.
- *Generic Interrupt Controller* (GIC) CPU interface to connect to an external interrupt distributor.
- Generic Timer interface supporting 64-bit count input from an external system counter.
- 40-bit or 48-bit configurable physical address width

Memory system features

- Separate L1 data cache and L1 instruction cache that are private to each core.
- Two optional *Tightly Coupled Memories* (TCMs) that are private to each core, an ITCM for instructions and literal pool data and a DTCM for data.
- An optional, shared (between all cores), and unified (instructions and data) L2 cache.
- Partial L2 cache powerdown support.
- A shared CHI.E or AXI5 256-bit *Main Manager* (MM) port for instruction and data access.
- An optional and shared AXI5 256-bit *Low-latency RAM* (LLRAM) port for instruction and data access.
- A shared ACE5-Lite 128-bit *Main Accelerator Coherency Port* (MACP) for external access to MM address ranges.
- A shared ACE5-Lite 128-bit *ACE-Lite Subordinate* (ACELS) port for external access to TCMs and also LLRAM port.
- An optional and shared AXI5 64-bit *Shared Peripheral Port* (SPP) for peripheral access.
- An optional and per-core AXI5 32-bit *Low-latency Peripheral Port* (LLPP) for peripheral access.
- Optional *Error Correcting Code* (ECC), *Single Error Correct Double Error Detect* (SECDED) or *Double Error Detect* (DED) protection for all of the instantiated cache tag and data RAMs and the TCM RAMs.

Debug features

- Arm®v8-R AArch64 debug logic with debug over powerdown support.
- *Reliability, Availability, and Serviceability* (RAS) Extension support.
- *Embedded Trace Macrocell* (ETM), compliant with ETMv4.5, for instruction and data trace.
- Performance Monitors Extension support for software profiling and performance debugging based on the PMUv3 architecture.
- *Cross Trigger Interface* (CTI) for multiprocessor debugging.

- Optional support for integrating CoreSight™ *Embedded Logic Analyzer*, ELA-600 for advanced debug capability and signal observability. The CoreSight™ ELA-600 is a separately licensable product.
- A production *Memory Built-In Self-Test* (MBIST) for testing memories at boot time.

1.3 Configuration options

You can configure the Cortex®-R82 processor during the rendering and integration stages to meet your functional requirements.

To configure the Cortex®-R82 processor, you must decide the values of:

- Configuration parameters when the Cortex®-R82 processor RTL is rendered before the implementation.
- Top-level configuration inputs when the Cortex®-R82 processor is integrated in the system after the implementation.



All top-level configuration inputs are sampled in the PERIPHCLK domain, just after the Cortex®-R82 processor comes out of Cold processor reset.

1.3.1 Configuration parameters

The configuration parameters affect the global and per-core RTL parameters.

Global parameters apply to the entire cluster while per-core parameters can be configured differently for different cores within the cluster.

Some memories and interfaces are optional so that you can configure the memory system according to your system requirements. When you choose to exclude:

- An optional memory:
 - For the *Instruction Tightly Couple Memory* (ITCM) and *Data Tightly Coupled Memory* (DTCM), the logic is removed.
 - For the L2 memory, the logic is always present but the RAM size is 0.
- An optional interface, the logic is removed.

The following table shows the main configuration parameters.



Warning

To successfully use a Cortex®-R82 processor core configured without Advanced SIMD and floating-point support (`NEON_FP<m> = 0`), you must ensure that software binaries running on this core do not use any Advanced SIMD or floating-point instructions. Advanced SIMD or floating-point instructions can be introduced in a number of ways. For example:

- By explicitly using them in assembly code.
- By using float or double data types or SIMD intrinsics in C or other high-level languages.
- By the compiler if Advanced SIMD/floating-point code generation is not disabled (such as auto-vectorization flags are enabled).
- By software libraries that are linked either explicitly by the user or implicitly by the compiler (independently of whether Advanced SIMD/floating-point code generation is enabled for the source being compiled). For further information, please refer to: <https://developer.arm.com/documentation/ka005579/latest>

Table 1-1: Configuration parameters

Parameter name	Scope	Permitted values	Description
NUM_CORES	Global	1, 2, 3, 4, 5, 6, 7, 8	Controls the number of physical cores
CHI	Global	0, 1	Controls the type of protocol the <i>Main Manager</i> (MM) interface implements 0 MM interface is AXI 1 MM interface is CHI
RAM_PROTECTION	Global	0, 1	Controls the inclusion of structures that are required to support internal RAM protection functionality 0 Not included 1 Included
ELA	Global	0, 1	Controls the inclusion of the CoreSight™ <i>Embedded Logic Analyzer</i> , ELA-600 0 Not included 1 Included Note: The ELA-600 is a separately licensable product
ELA_ATB_FIFO_DEPTH	Global	4, 8, 16, 32, 64	Configures the ELA-600 ATB FIFO depth in powers of two. See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual for more information.

Parameter name	Scope	Permitted values	Description
L2_CACHE_SIZE	Global	0, 96, 128, 192, 256, 384 512, 768, 1024, 1536, 2048, 3072, 4096	Controls the size of the L2 cache 0: L2 cache logic is present but RAM size is 0 96: 96KB cache 128: 128KB cache 192: 192KB cache 256: 256KB cache 384: 384KB cache 512: 512KB cache 768: 768KB cache 1024: 1024KB cache 1536: 1536KB cache 2048: 2048KB cache 3072: 3072KB cache 4096: 4096KB cache
L2_DATA_WR_LATENCY	Global	0, 1, 2	L2 cache data RAM input latency 0 1 cycle input delay from L2 data RAMs 1 2 cycles input delay from L2 data RAMs 2 2 cycles input delay plus 1 cycle hold from L2 data RAMs
L2_DATA_RD_LATENCY	Global	0, 1	L2 cache data RAM output latency 0 2 cycles output delay from L2 data RAMs 1 3 cycles output delay from L2 data RAMs
L2_DATA_RD_SLICE	Global	0, 1	L2 cache data RAM output register slice 0 No register slice 1 Register slice included
L2_DATA_STRETCH_CLK	Global	0, 1	L2 cache data RAMs clock pulse stretch 0 Not stretched 1 Stretched

Parameter name	Scope	Permitted values	Description
L2_SLICES	Global	1, 2	Controls the number of slices and RAM partitions the L2 cache implements 1 The L2 cache implements a single slice and a single RAM partition 2 The L2 cache implements two slices and two RAM partitions
CPU_SLICE	Global	0, 1	Controls whether extra register slices exist between the cores and the cluster logic. 0 No additional register slices exist between the cores and cluster logic 1 One additional register slice exists between the cores and cluster logic Note: Depending on your target frequency, setting CPU_SLICE to 1 may help with the timing closure.
PPU_RST_STATE	Global	0, 1	Power on reset state for the <i>Power Policy Units</i> (PPUs) 0 Cluster and all core PPUs reset to off 1 Cluster and all core PPUs reset to on
ACELS_ID_WIDTH	Global	Any integer value between 8 and 24	Defines the width of the <i>ACE-Lite Subordinate</i> (ACELS) port ID signals AWIDS, BIDS, ARIDS, and RIDS.
MACP_ID_WIDTH	Global	Any integer value between 8 and 24	Defines the width of the <i>Main Accelerator Coherency Port</i> (MACP) ID signals AWIDA, BIDA, ARIDA, and RIDA.
UB_ID_WIDTH	Global	Any integer value between 1 and 24	Defines the width of the <i>Utility bus</i> port ID signals AWIDU, BIDU, ARIDU, and RIDU.
LLPP	Global	0, 1	Controls the existence of cores' <i>Low-latency Peripheral Port</i> (LLPP) 0 No LLPP regions and ports exist. All related AXI inputs and outputs as well as the CFGLLPPIMP and CFGLLPPBASEADDR pins are rendered out by the configuration script 1 All cores include logic to support LLPP regions and ports. They can be enabled or disabled by the CFGLLPPIMP pin
SPP	Global	0, 1	Controls the existence of the cluster <i>Shared Peripheral Port</i> (SPP) 0 No SPP region and port exists. All related AXI inputs and outputs as well as the CFGSPPIMP and CFGSPPBASEADDR pins are rendered out by the configuration script 1 The processor includes logic to support the SPP region and port. It can be enabled or disabled by the CFGSPPIMP pin

Parameter name	Scope	Permitted values	Description
LLRAM	Global	0, 1	Controls the existence of the cluster <i>Low-latency RAM</i> (LLRAM) interface 0 No LLRAM region, port and coherency logic exists. All related AXI inputs and outputs as well as the CFGLLRAMIMP, CFGLLRAMEN and CFGLLRAMBASEADDR pins are rendered out by the configuration script 1 The processor includes logic to support the LLRAM region, port, and coherency. It can be enabled or disabled by the CFGLLRAMIMP pin and its behavior out of reset can be further defined by the CFGLLRAMEN pin
DENSE_CS_ADDR_MAP	Global	0, 1	Controls how CoreSight™ components are mapped in the Utility bus and the Debug port 0 Sparse memory map. Each component occupies 64KB. The Utility bus has 23-bit address signals. The Debug port has 24-bit address signals 1 Dense memory map. Each component occupies 4KB. The Utility bus has 17-bit address signals. The Debug port has 18-bit address signals
PA_W	Global	40, 48	Controls the Physical Address width 40: 40-bit physical address 48: 48-bit physical address
NUM_EL2_MPU_REGIONS<m>	Per-core	0, 16, 32	Controls the number of EL2 <i>Memory Protection Unit</i> (MPU) regions. <m> is the core number
NUM_EL1_MPU_REGIONS<m>	Per-core	0, 16, 32	Controls the number of EL1 MPU regions. <m> is the core number. Value 0 is only supported when the core instance includes support for <i>Virtual Memory System Architecture</i> (VMSA).
NEON_FP<m>	Per-core	0, 1	Controls the inclusion of Advanced SIMD and floating-point support. <m> is the core number. 0 No Advanced SIMD and no floating-point support included 1 Advanced SIMD and half-precision, single-precision, and double-precision floating-point functionality included
LOW_LATENCY_SP<m>	Per-core	0, 1	Controls the inclusion of an additional single-precision floating-point pipeline which has lower result latencies than the default floating-point pipeline. <m> is the core number. 0 Default floating-point pipeline only 1 Low-latency single-precision pipeline included
VMSA<m>	Per-core	0, 1	Controls the inclusion of VMSA functionality. <m> is the core number. 0 Not included 1 Included

Parameter name	Scope	Permitted values	Description
L1_ICACHE_SIZE<m>	Per-core	16, 32, 64, 128	Controls the size of the L1 instruction cache. <m> is the core number. 16: 16KB cache 32: 32KB cache 64: 64KB cache 128: 128KB cache
L1_DCACHE_SIZE<m>	Per-core	16, 32, 64	Controls the size of the L1 data cache. <m> is the core number. 16: 16KB cache 32: 32KB cache 64: 64KB cache
ITCM_SIZE<m>	Per-core	0, 16, 32, 64, 128, 256, 512, 1024	Controls the size of the <i>Instruction Tightly Coupled Memory</i> (ITCM). <m> is the core number. 0: ITCM logic is removed and RAM size is 0 16: 16KB cache 32: 32KB cache 64: 64KB cache 128: 128KB cache 256: 256KB cache 512: 512KB cache 1024: 1024KB cache
ITCM_WAIT<m>	Per-core	0, 1, 2, 3	Controls the number of wait states that are incurred by accesses to the ITCM. <m> is the core number.
ITCM_STRETCH_CLK<m>	Per-core	0, 1	ITCM RAMs clock pulse stretch. <m> is the core number. 0 Not stretched 1 Stretched

Parameter name	Scope	Permitted values	Description
DTCM_SIZE<m>	Per-core	0, 16, 32, 64, 128, 256, 512, 1024	Controls the size of the <i>Data Tightly Coupled Memory</i> (DTCM). <m> is the core number. 0: DTCM logic is removed and RAM size is 0 16: 16KB cache 32: 32KB cache 64: 64KB cache 128: 128KB cache 256: 256KB cache 512: 512KB cache 1024: 1024KB cache
DTCM_WAIT<m>	Per-core	0, 1, 2, 3	Controls the number of wait states that are incurred by accesses to the DTCM. <m> is the core number.
DTCM_STRETCH_CLK<m>	Per-core	0, 1	DTCM RAMs clock pulse stretch. <m> is the core number. 0 Not stretched 1 Stretched

1.3.2 Integration-time configuration options

The following table shows the integration-time configuration options that you can configure by setting the top-level inputs.



Note

In addition to the features in the following table, you can also configure the *Main Manager* (MM) and *Low-latency RAM* (LLRAM) broadcast signal behavior as defined by the AMBA® specification. See *Arm® Cortex®-R82 Processor Configuration and Integration Manual* for more information on the Cortex®-R82 processor signals.

Table 1-2: Integration-time configuration options

Feature	Scope	Permitted values	Description
<i>Low-latency RAM</i> (LLRAM) manager interface	Global	0, 1	Implemented with CFGLLRAMIMP 0 Not implemented. 1 Implemented.
LLRAM base address	Global	Aligned to 256MB	Configured with CFGLLRAMBASEADDR[39:28]

Feature	Scope	Permitted values	Description
LLRAM enable out of reset	Global	0, 1	Configured with CFGLLRAMEN 0 LLRAM disabled out of reset. 1 LLRAM enabled out of reset.
LLRAM shared	Global	0, 1	Configured with CFGLLRAMSHARED. If set to 1, it indicates that there are multiple clusters connected to the same LLRAM and share data across the cluster. 0 LLRAM not shared. 1 LLRAM shared. Caution: This feature is not available in the Cortex®-R82 processor. You must tie CFGLLRAMSHARED LOW.
Poison support for <i>Main Manager</i> (MM) port	Global	0, 1	Configured with CFGMMPOISON 0 MM port does not support poison. 1 MM port supports poison.
<i>Low-latency Peripheral Port</i> (LLPP) manager interface	Global	0, 1	Implemented with CFGLLPPIMP 0 Not implemented. 1 Implemented.
LLPP base address	Global	Aligned to 128MB	Configured with CFGLLPPBASEADDR[39:27]
<i>Shared Peripheral Port</i> (SPP) manager interface	Global	0, 1	Implemented with CFGSPPIMP 0 Not implemented. 1 Implemented.
SPP base address	Global	Aligned to 128MB	Configured with CFGSPPBASEADDR[39:27]
Base address of the TCMs on the <i>ACE-Lite Subordinate</i> (ACELS) interface	Global	Aligned to 16MB	Configured with CFGACELSTCMBASEADDR[39:24]
Value of Multiprocessor Affinity Register affinity level 2	Global	-	Configured with CFGMPIDRAFF2[7:0]
Value of Multiprocessor Affinity Register affinity level 3	Global	-	Configured with CFGMPIDRAFF3[7:0]
GIC CPU interface disable	Global	0, 1	Configured with GICCDISABLE 0 GIC CPU interface enabled. 1 GIC CPU interface disabled.

Feature	Scope	Permitted values	Description
RAM protection enable out of reset	Global	0, 1	Configured with CFGGRAMPROTEN 0 RAM protection disabled out of reset 1 RAM protection enabled out of reset
L2 cache POP EVA feature	Global	0, 1	Configured with CFGGL2EVAIMP 0 L2 cache data RAMs do not implement the POP Eviction-Allocation optimization 1 L2 cache data RAMs implement the POP Eviction-Allocation optimization, and this is enabled out of reset
ITCM base address	Per-core	Size-aligned to ITCM_SIZE<m>	Configured with CFGITCMBASEADDRm[39:14] where m is the core number from 0 to NUM_CORES-1.
ITCM enable out of reset	Per-core	0, 1	Configured with CFGITCMENm where m is the core number from 0 to NUM_CORES-1. 0 ITCM disabled out of reset. 1 ITCM enabled out of reset.
DTCM base address	Per-core	Size-aligned to DTCM_SIZE<m>	Configured with CFGDTCMBASEADDRm[39:14] where m is the core number from 0 to NUM_CORES-1.
Data endianness	Per-core	0, 1	Configured with CFGENDm where m is the core number from 0 to NUM_CORES-1. Controls the out of reset value of the SCTLR_EL2.EE bit. 0 Little-endian. 1 Byte-invariant big-endian.
Reset vector base address	Per-core	Aligned to 4B	Configured with CFGRVBARADDRm[39:2] where m is the core number from 0 to NUM_CORES-1. Controls the out of reset value of the RVBAR_EL2.RVBARADDR field.

1.4 Supported standards and specifications

The Cortex®-R82 processor complies with, or implements, the relevant Arm architectural standards and protocols, and relevant external standards.

This book complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources except for some system register content.

Arm architecture

The Cortex®-R82 processor implements the Arm®v8-R AArch64 architecture. This includes:

- AArch64 Execution state only. There is no support for AArch32 Execution state.
- Support for Exception levels EL0, EL1, and EL2. There is no support for EL3.
- A64 instruction set for the Arm®v8-R AArch64 architecture.
- Advanced SIMD and floating-point functionality that complies with ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

Bus architecture

The Cortex®-R82 processor is compliant with the following:

- AMBA 5 CHI (Issue E). See the [AMBA® 5 CHI Architecture Specification](#).
- AMBA 5 AXI and ACE protocol (Issue H). See the [AMBA® AXI Protocol Specification](#).
- AMBA AXI5-Stream protocol (Issue B). See the [AMBA® AXI-Stream Protocol Specification](#).
- AMBA APB5 protocol (Issue D). See the [AMBA® APB Protocol Specification](#).
- AMBA ATB protocol (Issue C). See the [Arm® AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1](#).

Generic Interrupt Controller (GIC) architecture CPU interface

The Cortex®-R82 processor supports the GICv3.2 architecture.

See the [Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4](#).

Generic Timer architecture

The Cortex®-R82 processor implements the Arm® Generic Timer architecture.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

Debug architecture

The Cortex®-R82 processor implements the Arm®v8-R AArch64 debug architecture including debug features up to Arm®v8.4 architecture, the Armv8.3-DoPD extension, and the Arm®v8.3 debug over powerdown support.

For more information, see the:

- [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).
- [Arm® CoreSight™ Architecture Specification v3.0](#).
- [Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual](#).

Embedded Trace Macrocell (ETM) architecture

The Cortex®-R82 processor implements the ETMv4.5 architecture.

See the [Arm® Embedded Trace Macrocell Architecture Specification ETMv4.0 to ETM4.6](#) for more information.

Performance Monitoring Unit (PMU)

The Cortex®-R82 processor implements the PMUv3 architecture with the Arm®v8.4 PMU extension.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

Reliability, Availability, and Serviceability (RAS)

The Cortex®-R82 processor implements the Arm®v8.4 RAS extension.

See the [Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability \(RAS\), for A-profile architecture](#) for more information.

1.5 Design process

The Cortex®-R82 processor is delivered as a synthesizable RTL description in SystemVerilog *Hardware Description Language* (HDL). Before the Cortex®-R82 processor can be used in a product, it must go through the following processes:

Implementation

The implementer configures and synthesizes the RTL to produce a hard macrocell. This process includes integrating the cache, TCM, duplicate tag, and branch predictor RAMs into the design.

Integration

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

Programming

In the final process, the system programmer develops the software to configure and initialize the Arm processor and tests the application software.

Each process can be performed by a different party. Implementation and integration choices affect the behavior and features of the Cortex®-R82 processor.

The operation of the final device depends on the following:

Build configuration

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

Configuration inputs

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

Software configuration

The programmer configures the processor by programming particular values into registers. These configuration choices affect the behavior of the processor.

1.6 Documentation

The Cortex®-R82 processor documentation describes the functionality of the processor and explains how to configure, integrate, and implement it.

The Cortex®-R82 processor documentation includes a Technical Reference Manual and a Configuration and Integration Manual.

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the key features of the processor. It is a helpful reference at all stages of the design flow. Some behavior described in the TRM might not be relevant because of the particular way that the Cortex®-R82 processor is implemented and integrated. If you are programming the Cortex®-R82 processor, you need additional information from:

- The implementer about the build configuration of the implementation.
- The integrator about the signal configuration of the device that you are using.

Configuration and Integration Manual

The *Configuration and Integration Manual* (CIM) describes the Cortex®-R82 processor deliverables and how to use them to perform implementation and integration of the processor.

The *Configuration and Integration Manual* (CIM) describes:

- How to configure the RTL source files with the build configuration options.
- How to integrate RAM arrays.
- How to validate the RTL.
- How to run delivered tests.
- Considerations for floorplanning.
- How to integrate the processor into a SoC. This includes describing the signals that must be tied off to configure the macrocell.
- The integration kit description.
- The processes to sign off the configured design.

If you are integrating an already-implemented macrocell of the Cortex®-R82 processor you need additional information from the implementer about the build configuration of the implementation.

The Arm product deliverables include reference scripts and information on how to use them to implement your design. The methodology flows supplied by Arm are example reference implementations. For EDA tool support, contact your EDA vendor.

The CIM is a confidential manual that is available only to licensees.

1.7 Product revisions

The following product revisions have been released.

r1p0

First limited access release for r1p0

r1p1

First early access release for r1p1

r2p0

First early access release for r2p0

r3p0

First early access release for r3p0

r3p1

First release for r3p0

2. Technical overview

This chapter describes the Cortex®-R82 processor components and interfaces.

2.1 Terminology

In this manual, the following terms refer to the descriptions that are provided below.

Core

A core includes all the logic related to the data processing unit, memory system and management, power management, and core-level debug and trace logic. In the context of the Cortex®-R82 processor, CPU and core are used interchangeably.

In this manual, NUM_CORES refers to the number of cores within the Cortex®-R82 processor and <m> refers to the core instance number.

Cluster

In the context of the Cortex®-R82 processor, the cluster refers to the *CPU Bridge (System side)* (CBS), the *Shared Bridge* (SB), all cores, and the logic that is shared among cores. Shared logic includes the *CPU Bridge (CPU side)* (CBC), the L2 cache, and the coherency logic that maintains coherency between the caches in the cores and the L2 cache and the *Low-latency RAM* (LLRAM) memory. There is also shared debug logic at the cluster level.

Processor

In the context of the Cortex®-R82 processor, the processor is the top-level unit that contains the cluster and the *Power Policy Units* (PPUs).

DebugBlock

DebugBlock is a dedicated debug component separate from the Cortex®-R82 processor. The DebugBlock is instantiated as a separate top-level unit to allow you to implement the debug components in an always On power domain. Although instantiated as a separate unit, the DebugBlock still forms part of the Cortex®-R82 processor.

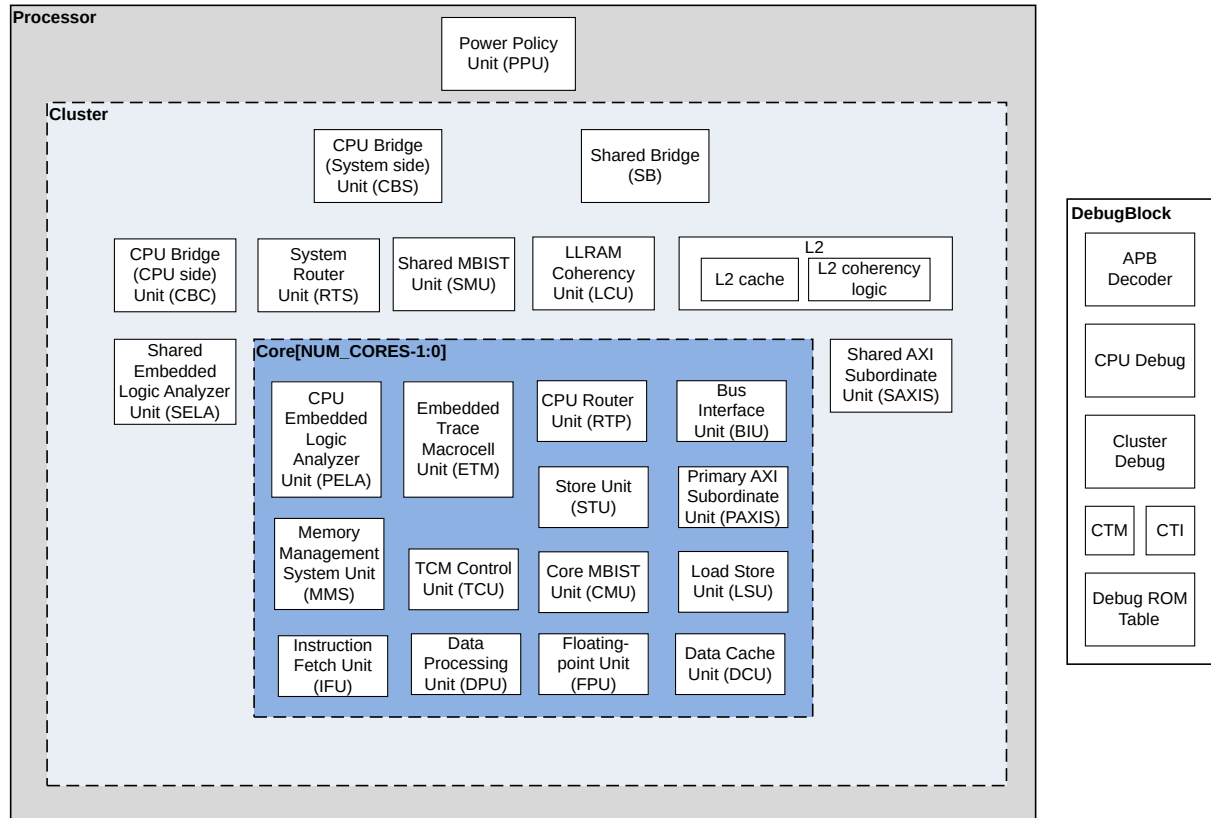
2.2 Components

The Cortex®-R82 processor system includes two top-level modules:

- The Cortex®-R82 processor
- The DebugBlock

The DebugBlock is separated from the Cortex®-R82 processor to allow you to implement the debug components in an always On power mode, enabling debug over powerdown.

The following figure shows the main components of the Cortex®-R82 processor.

Figure 2-1: Functional block diagram

Power Policy Unit

The Cortex®-R82 processor includes several *Power Policy Units* (PPUs) that control power modes and resets. The PPU can be programmed to directly select a specific power mode or to autonomously switch between power modes within a specified range, based on the requirements of the processor.

The PPU can be programmed using the Utility bus, either by a *System Control Processor* (SCP) or by the Cortex®-R82 processor (through a loopback connection).

For more information on the PPUs, see [6. Power and reset control with PPUs](#) on page 80.

Shared Bridge

The *Shared Bridge* (SB) decouples the DebugBlock and other components in the system from the CPU bridge in each core. The SB includes clock and power control logic for the cluster and interacts with the L2 coherency logic for SCLK clock gating.

CPU Bridge (System side) Unit

There is one *CPU bridge (System side) Unit* (CBS) in the cluster. The CPU bridge controls buffering between the cores and the Cortex®-R82 processor.

CPU Bridge (CPU side) Unit

There is one *CPU Bridge (CPU side) Unit* (CBC) for all the cores in the cluster. The CPU bridge controls buffering between the cores and the Cortex®-R82 processor.

System Router Unit

The *System Router Unit* (RTS) is instantiated in the cluster as a single unit. The RTS controls the packet traffic at the cluster level such as routing the *Generic Interrupt Controller* (GIC) packets between the SB in the cluster and the *CPU Router Units* (RTPs) within the cores.

Shared MBIST Unit

The *Shared MBIST Unit* (SMU) provides production MBIST for the *LLRAM Coherency Unit* (LCU) and L2 cache RAMs.

LLRAM Coherency Unit

The *LLRAM Coherency Unit* (LCU) provides coherent access to the *Low-latency RAM* (LLRAM) port for up to eight cores. The LCU also provides coherent access to LLRAM port for an external agent that needs I/O coherency with the Cortex®-R82 processor. The LCU also provides access to the *Shared Peripheral Port* (SPP).

L2

The L2 consists of the L2 cache RAMs and the L2 coherency logic. The L2 is required to interface the cores to an AXI or a CHI interconnect.

For more information on the L2 memory system, see [8.4 L2 memory system](#) on page 120.

L2 cache

The L2 cache is unified (it can cache both instructions and data) and shared by all the cores in the cluster. The L2 cache is 8-way, set-associative with a configurable size of 0KB, 128KB, 256KB, 512KB, 1MB, 2MB, or 4MB. Cache lines have a fixed length of 64 bytes.

L2 coherency logic

The L2 coherency logic maintains coherency between all the cores and caches within the cluster for the *Main Manager* (MM) port accesses.

When the MM port is configured to be CHI and the cluster is connected to another coherent cluster through a coherent interconnect, the L2 coherency logic also automatically maintains coherency with that other cluster.

The L2 coherency logic contains buffers that can handle direct cache-to-cache transfers between cores without having to read or write data to the L2 cache. Cache line migration enables dirty cache lines to be moved between cores. There is no requirement to write back transferred cache line data to the L2 cache.

Shared AXI Subordinate Unit

The *Shared AXI Subordinate Unit* (SAXIS) enables external read and write access to the *ACE-Lite Subordinate* (ACELS) port. These reads and writes are then routed to the *Primary AXI Subordinate Unit* (PAXIS) or the LCU.

Shared Embedded Logic Analyzer Unit

The *Shared Embedded Logic Analyzer Unit* (SELA) provides support for a cluster level CoreSight™ ELA-600 *Embedded Logic Analyzer* to monitor the signals related to the shared logic.

CPU Embedded Logic Analyzer Unit

The *CPU Embedded Logic Analyzer Unit* (PELA) provides support for per-core CoreSight™ ELA-600 to monitor the signals relate to the core. The configuration option to pre-integrate ELAs is global.



The CoreSight™ ELA-600 is a separately licensable product.

Embedded Trace Macrocell Unit

Each core has an *Embedded Trace Macrocell* (ETM) unit that enables per-core ETM instruction and data trace on separate ATB buses:

- An ATB4 32-bit bus for instruction.
- An ATB4 128-bit bus for data and ELA.

The trace is generated per-core but the ATB buses are shared between the cores.

For more information on the ETM, see [15. ETM](#) on page 267.

CPU Router Unit

There is one *CPU Router Unit* (RTP) that is instantiated within each core. The RTPs are connected to the RTS in the cluster.

Each RTP controls the packet traffic within the core such as routing packages between the RTP and the *Data Processing Unit* (DPU) in the core.

Bus Interface Unit

The *Bus Interface Unit* (BIU) is responsible for driving the L2 and LCU read and write interfaces. The BIU receives the read and write requests from the *Load Store Unit* (LSU), *Data Cache Unit* (DCU), *Translation Lookaside Buffer* (TLB), *Instruction Fetch Unit* (IFU), and *Store Unit* (STU).

Store Unit

The *Store Unit* (STU) merges and forwards (as appropriate) stores to *Instruction Tightly Coupled Memory* (ITCM), *Data Tightly Coupled Memory* (DTCM), L1 caches, *Low-latency Peripheral Port* (LLPP), *Shared Peripheral Port* (SPP), *Low-latency RAM* (LLRAM), and *Main Manager* (MM).

Primary AXI Subordinate Unit

The *Primary AXI Subordinate Unit* (PAXIS) enables the read and write access to the *Tightly Coupled Memories* (TCMs). The reads and writes both directly route to the *TCM Unit* (TCU) from the PAXIS.

Memory Management System Unit (MMS)

The Cortex®-R82 processor implements *Protected Memory System Architecture* (PMSA) and optional *Virtual Memory System Architecture* (VMSA).

Hypervisor software running at EL2 selects between PMSA and VMSA on a per-operating system basis.

Memory Protection Unit

The *Memory Protection Unit* (MPU) implements the PMSA and determines the attributes for each memory location including permissions, type, and cacheability. Two programmable MPUs are provided, controlled from EL1 and EL2 respectively.

Access permissions determine which levels of privilege are permitted to access a location and whether write access or instruction execution are permitted. Memory type and cacheability affect how the Cortex®-R82 processor handles particular accesses, for example, if the processor permits two stores to be merged into a single write access. These attributes and their meanings are defined by the Arm architecture.

Memory Management Unit

The *Memory Management Unit* (MMU) implements the VMSA and provides memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables. This information is cached in the *Translation Lookaside Buffer* (TLB) when an address is translated. The TLB entries include global and *Address Space Identifiers* (ASIDs) to prevent context switch TLB flushes. They also include *Virtual Machine Identifiers* (VMIDs) to prevent TLB flushes on virtual machine switches by the hypervisor.

For more information on the memory management, see [9. Memory management](#) on page 195.

TCM Control Unit

The *Tightly Coupled Memory Control Unit* (TCU) is responsible for arbitration between all requests to the *Instruction Tightly Coupled Memories* (ITCMs) and *Data Tightly Coupled Memories* (DTCMs). The TCU contains two arbitration pipelines for managing requests to ITCMs and DTCMs.

Each core within the Cortex®-R82 processor has:

- An optional *Instruction Tightly Coupled Memory* (ITCM) with configurable size 0 or from 16KB to 1MB in powers of 2. ITCM provides lowest-latency access for instructions and data. Optional here means that the logic is always present but the size can be 0.



Note

ITCM is unified, that is, although it is optimized for instruction use, it is also available for data.

-
- An optional *Data Tightly Coupled Memory* (DTCM) with configurable size 0 or from 16KB to 1MB in powers of 2. DTCM provides lowest-latency access for data only. Optional here means that the logic is always present but the size can be 0.



DTCM is for data only and it cannot be used to fetch instructions from.

For more information on the TCMs, see [8.2 TCM memories](#) on page 106.

Core MBIST Unit

The *Core MBIST Unit* (CMU) provides production MBIST for the core RAMs.

Load Store Unit

The *Load Store Unit* (LSU) is responsible for execution of all instructions that affect the L1 data memory system.

Instruction Fetch Unit

The *Instruction Fetch Unit* (IFU) speculatively fetches instructions from the *Instruction Tightly Coupled Memory* (ITCM), L1 instruction cache, or the main memory.

The IFU predicts the outcome of branches in the instruction stream and passes the instructions to the *Data Processing Unit* (DPU) for processing.

Data Processing Unit

The *Data Processing Unit* (DPU) includes the instruction decoders, the integer execution pipelines, and the control logic of the Cortex®-R82 processor. It receives instructions from the IFU. The DPU executes the integer instructions and works in conjunction with the FPU and LSU to execute FP/SIMD instructions and instructions which require data transfer to or from the memory system.

The DPU interfaces with IFU, LSU, STU, DCU, TCU, LCU, RTP, ETM, and CBC.

The DPU implements the *Generic Interrupt Controller* (GIC) CPU interface as well as the *Performance Monitoring Unit* (PMU).

Floating-point Unit

The *Floating-point Unit* (FPU) is responsible for decoding and executing the Advanced SIMD and floating-point instructions.

Supporting Advanced SIMD and floating-point instructions is optional and can be configured separately per-core by the `NEON_FP` parameter.

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 half-precision, single-precision, and double-precision floating-point operations.

**Note**

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

For more information on the Advanced SIMD and floating-point support, see [16. Advanced SIMD and floating-point support](#) on page 277.

Data Cache Unit

The *Data Cache Unit* (DCU) is responsible for all operations accessing the L1 Data cache. The DCU arbitrates between the requests and is responsible for responding to snoop requests to maintain coherency.

DebugBlock

The DebugBlock transfers trigger events to/from the Cortex®-R82 processor.

The DebugBlock is separated from the Cortex®-R82 processor to facilitate the following system design options:

- The DebugBlock is placed in a separate power domain, to ensure that it is possible to maintain the connection to a debugger while the cores and cluster are powered down.
- The DebugBlock is physically placed with the other CoreSight logic in the SoC, rather than close to the cluster.

The separate power domains allow the cores and the cluster to be powered down while maintaining essential state that is required to continue debugging. Separating the logical power domains into physical domains is optional and might not be available in individual systems.

APB Decoder

The APB Decoder is responsible for gathering the external CoreSight component signals such as *Debug Access Port* (DAP) inputs and the debug events from the Cortex®-R82 processor. It then generates the internal select signals to activate the appropriate internal component within the DebugBlock.

CPU Debug

The CPU Debug handles all core related accesses from both the external CoreSight component and the Cortex®-R82 processor and generates all the necessary transactions to a core within the Cortex®-R82 processor.

There is one CPU Debug module for each core within the Cortex®-R82 processor.

Cluster Debug

The Cluster Debug handles all cluster related accesses from both the external CoreSight component and the Cortex®-R82 processor and generates all the necessary transactions to relevant cluster components such as the cluster PMU and cluster ELA.

There is one Cluster Debug module for all the cluster components.

CTI and CTM

The DebugBlock implements *Embedded Cross Trigger* (ECT). A *Cross Trigger Interface* (CTI) is allocated to each core within the Cortex®-R82 processor. An additional CTI is allocated to the cluster PMU and, if present, to the cluster ELA.

The CTIs enable the debug logic, ETM, and other CoreSight components to interact with each other.

The CTIs are interconnected through the *Cross Trigger Matrix* (CTM). A single external channel interface is implemented to allow cross-triggering to be extended to the SoC.

Debug ROM Table

The Debug ROM table contains a list of components in the system. Debuggers can use the Debug ROM table to determine which CoreSight components are implemented.

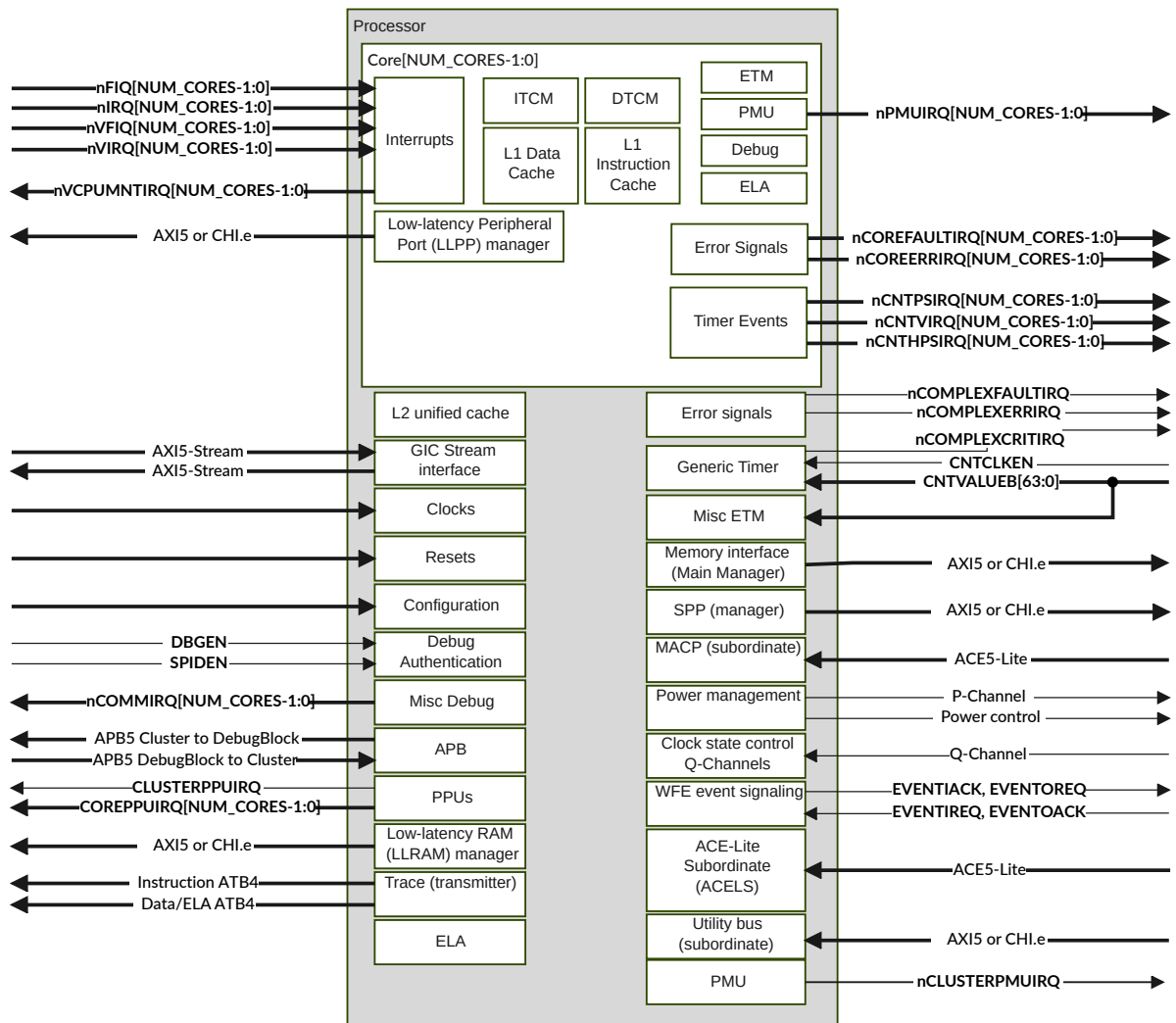
2.3 Interfaces

The Cortex®-R82 processor has several interfaces to connect it to a SoC.

The following figure shows the interfaces of the Cortex®-R82 processor.



NUM_CORES is the number of physical cores from 1 to 8.



The following table describes the interfaces of the Cortex®-R82 processor.

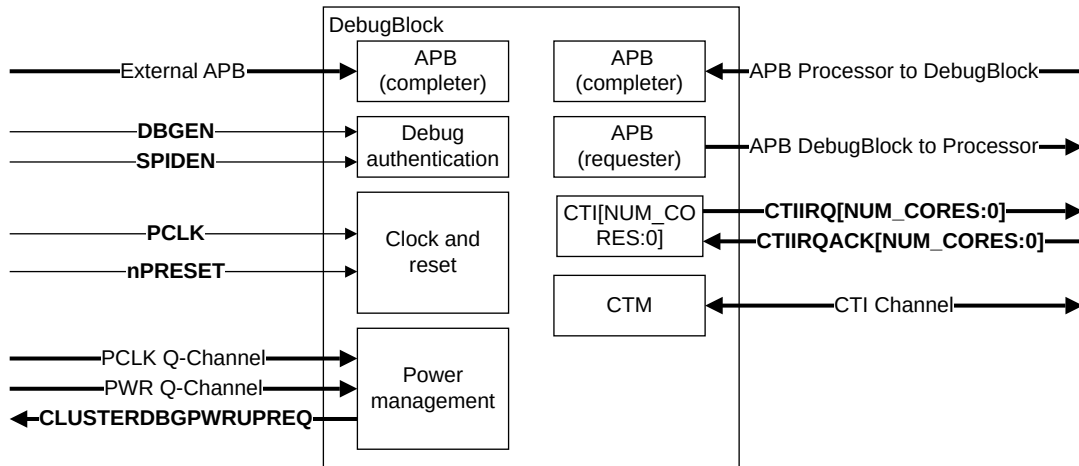
Table 2-1: Processor interfaces

Purpose	Protocol	Notes
Low-latency Peripheral Port (LLPP)	AMBA® AXI5 (Issue H) 32-bit	Each core within the Cortex®-R82 processor has an optional private LLPP manager for minimum latency access to memory and devices outside the cluster.
Generic Interrupt Controller (GIC) Stream interface (processor to GIC)	AMBA® AXI5-Stream (Issue B) 32-bit	AXI5-Stream interface for interrupts from the Cortex®-R82 processor to the GIC.
GIC Stream interface (GIC to processor)	AMBA® AXI5-Stream (Issue B) 32-bit	AXI5-Stream interface for interrupts from the GIC to the Cortex®-R82 processor.

Purpose	Protocol	Notes
Cortex®-R82 processor to DebugBlock	AMBA® APB5 (Issue D) 32-bit	APB interface from the Cortex®-R82 processor to the DebugBlock.
DebugBlock to Cortex®-R82 processor	AMBA® APB5 (Issue D) 32-bit	APB interface from the DebugBlock to the Cortex®-R82 processor.
Low-latency RAM (LLRAM)	AMBA® AXI5 (Issue H) 256-bit	The Cortex®-R82 processor has an optional LLRAM for low latency access to memory shared between cores within the cluster.
Trace	AMBA® ATB4 (Issue C) 32-bit instruction ATB trace 128-bit data ATB trace	The Cortex®-R82 processor has two transmitter ATB interfaces for instructions and data. An instruction trace funnel funnels all the processor ETM instruction trace streams into a single ATB trace bus. A data trace funnel funnels all the processor ETM data trace streams and all the ELA trace streams into a single ATB trace bus.
Utility bus	AMBA® AXI5 (Issue H) 64-bit	The Cortex®-R82 processor has a subordinate Utility bus that manages power states and provides access to <i>Power Policy Units</i> (PPUs) registers and <i>Reliability, Availability, and Serviceability</i> (RAS) registers in each core and the cluster.
ACE-Lite Subordinate (ACELS)	AMBA® ACE5-Lite (Issue H) 128-bit	The Cortex®-R82 processor has a subordinate ACELS bus that enables external agents to access to the TCMs and LLRAM port.
WFE event signaling	-	Signals for <i>Wait For Event</i> (WFE) wake-up events.
Clock state control	Q-Channel	Q-Channels for clock gating control.
Power state control	P-Channel	P-Channels for Cortex®-R82 processor power management.
Main Accelerator Coherency Port (MACP)	AMBA® ACE5-Lite (Issue H) 128-bit	The Cortex®-R82 processor has a subordinate MACP that provides access to the MM port to external managers.
Shared Peripheral Port (SPP)	AMBA® AXI5 (Issue H) 64-bit	The Cortex®-R82 processor has an optional shared SPP manager for minimum latency access to memory and devices.
Main Manager (MM)	AMBA® AXI5 (Issue H) or AMBA® CHI (Issue E)	The Cortex®-R82 processor has a configurable shared MM port for accesses to high-latency memory and non-critical peripherals.
Power management	P-Channel and Q-Channel	Cortex®-R82 enables communication to an external power controller for clock gating and powerdown.
Generic Timer	-	Input for the Generic Timer counter value. The counter value is distributed to all cores. Each core outputs timer events.
Design for Test (DFT)	-	Interface to support access for <i>Automatic Test Pattern Generation</i> (ATPG) scan-path testing.

DebugBlock interfaces

The following figure shows the interfaces of the DebugBlock.

Figure 2-3: DebugBlock interfaces

The following table describes the external interfaces of the DebugBlock.

Table 2-2: DebugBlock interfaces

Purpose	Protocol	Notes
External APB	AMBA® APB5 (Issue D)	Completer interface to external debug component, for example a <i>Debug Access Port</i> (DAP). Allows access to Debug registers and resources.
Cortex®-R82 processor to DebugBlock	AMBA® APB5 (Issue D)	APB interface from the Cortex®-R82 processor to the DebugBlock.
DebugBlock to Cortex®-R82 processor	AMBA® APB5 (Issue D)	APB interface from the DebugBlock to the Cortex®-R82 processor.
Cross-trigger channel interface	CTI	Allows cross-triggering to be extended to external SoC components.
Power management	Q-Channel	Enables communication to an external power controller to control clock gating and powerdown.

3. Programmers' model

This chapter provides a brief description of the Arm®v8-R AArch64 architecture for programmers.

For a complete description of the programmers' model, refer to the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

3.1 About the programmers' model

The Cortex®-R82 processor implements the Arm®v8-R AArch64 architecture. This includes:

- AArch64 Execution state only. There is no support for AArch32 Execution state.
- Support for Exception levels EL0, EL1, and EL2. There is no support for EL3.
- Secure state operation only at all Exception levels using the Secure-EL2 security model. There is no support for Non-secure state at any EL.
- Full implementation of the Arm®v8-A A64 instruction set with Arm®v8-R AArch64 *Instruction Set Architecture* (ISA) extensions.
- *Protected Memory System Architecture* (PMSA) at EL1 and EL2.
- *Optional Virtual Memory System Architecture* (VMSA) only at EL1.
- Arm®v8-R AArch64 debug architecture including debug features up to Arm®v8.4 architecture and the debug over powerdown support.
- *Generic Interrupt Controller* (GIC) with the GICv3.2 architecture.
- Advanced SIMD and floating-point operations in the A64 instruction set.

3.2 Arm®v8-R AArch64 architecture concepts

The following sections provide an introduction to the main architectural concepts and terminology used throughout the rest of this document.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.



An understanding of the terminology defined in this section is a prerequisite for understanding the remainder of this manual.

3.2.1 Architecture requirements

The Cortex®-R82 processor supports all the mandatory features from the Arm®v8.4 architecture and some optional features from the Arm®v8.5 and Armv8.6 architectural extensions.

The following table shows the architectural features that are implemented by the Cortex®-R82 processor.

Table 3-1: Architectural features implemented by the processor

Feature name	Description	Implemented by the processor
FEAT_GICv3	Generic Interrupt Controller v3	Yes
FEAT_GICv3p1	Generic Interrupt Controller v3.1	Yes
FEAT_GICv3_NMI	Non-maskable interrupts	No
FEAT_GICv3_LEGACY	Support for GICv2 legacy operation	No
FEAT_GICv3_TDIR	Trapping EL1 writes to ICV_DIR	Yes
FEAT_GICv4	Generic Interrupt Controller v4	No
FEAT_GICv4p1	Generic Interrupt Controller v4.1	No
FEAT_PMUv3	PMU extension	Yes
FEAT_ETMv4	Embedded Trace Macrocell v4.0	Yes
FEAT_ETMv4p1	Embedded Trace Macrocell v4.1	Yes
FEAT_ETMv4p2	Embedded Trace Macrocell v4.2	Yes
FEAT_ETMv4p3	Embedded Trace Macrocell v4.3	Yes
FEAT_ETMv4p4	Embedded Trace Macrocell v4.4	Yes
FEAT_ETMv4p5	Embedded Trace Macrocell v4.5	Yes
FEAT_ETMv4p6	Embedded Trace Macrocell v4.6	No
FEAT_RAS	Reliability, Availability and Serviceability extension	Yes
FEAT_SPE	Statistical Profiling Extension	No
FEAT_SVE	Scalable Vector Extension	No
FEAT_AMUv1	Activity Monitors	No
FEAT_MPAM	Memory Partitioning and Monitoring	No
FEAT_PCSRv8	PC Sample-based Profiling extension	Yes
FEAT_SHA1	SHA1 and SHA2-256 functionality	No
FEAT_SHA256		
FEAT_AES	Advanced Encryption Standard	No
FEAT_PMULL		
FEAT_DoubleLock	Double Lock	No
FEAT_SSBS	Speculative Store Bypass Safe	Yes
FEAT_SSBS2		
FEAT_CSV2	Cache Speculation Variant 2: Implementation of CSV2 without SCXTNUM registers	Yes
FEAT_CSV2_1p1		Yes
FEAT_CSV2_1p2		No
FEAT_CSV2_2		No

Feature name	Description	Implemented by the processor
FEAT_CSV3	Cache Speculation Variant 3	Yes
FEAT_SB	Speculation Barrier	Yes
FEAT_SPECRES	Prediction Invalidation	Yes
FEAT_CP15SDISABLE2	Prevents writes to a set of Secure CP15 registers	No
FEAT_DGH	Data Gathering Hint	Yes
FEAT_ETS	Enhanced Translation Synchronization	No
FEAT_nTLBPA	Intermediate caching of translation table walks	Yes (nTLBPA set to 1)
FEAT_CRC32	CRC32 Instructions	Yes
FEAT_LSE	Large System Extensions	Yes
FEAT_RDM	Rounding Double Multiply Accumulate	Yes, when NEON_FPM == 1
FEAT_HPDS	Hierarchical Permission Disables	Yes, when VMSAm == 1
FEAT_VHE	Virtualization Host Extensions	No
FEAT_PAN	Privileged Access-Never	Yes
FEAT_PAN3	Support for SCTLR_ELx.EPAN	No
FEAT_LOR	Limited Ordering Regions	No
FEAT_HAFDBS	Translation Table Hardware Management	Yes, when VMSAm == 1
FEAT_VMID16	16-bit VMID	No
FEAT_PMUV3p1	Arm v8.1 PMU extensions	Yes
FEAT_TTCNP	Translation Table Common Not Private translations	Yes, when VMSAm == 1
FEAT_XNX	Translation Table Stage 2 Unprivileged Execute-Never	Yes
FEAT_UAO	PSTATE override of Unprivileged Load/Store	Yes
FEAT_PAN2	AT S1E1R and AT S1E1W instruction variants	Yes
FEAT_DPB	Data Cache clean to Point of Persistence	Yes
FEAT_Debugv8p2	Arm v8.2 Debug extensions	Yes
FEAT_ASMv8p2	Arm v8.2 changes to the A64 Instruction Set Architecture	Yes
FEAT_IESB	Implicit Error Synchronization Barrier	Yes
FEAT_AA32HPD	AArch32 Hierarchical Permission Disables	No
FEAT_HPDS2	Translation Table Page Based Hardware Attributes	No
FEAT_LSMAOC	Load/Store Multiple Atomicity and Ordering Controls	No
FEAT_FP16	Half-precision floating-point data processing	Yes, when NEON_FPM == 1
FEAT_LVA	Large VA support	No
FEAT_LPA	Large PA and IPA support	No
FEAT_VPIPT	VMID-aware PIPT instruction cache	No
FEAT_PCSRv8p2	Arm v8.2 PC Sample-based Profiling extension	Yes

Feature name	Description	Implemented by the processor
FEAT_DotProd	Dot Product instructions	Yes, when NEON_FPm == 1
FEAT_FHM	Floating-point Half-precision Multiplication instructions	Yes, when NEON_FPm == 1
FEAT_SHA512	Cryptography support for SHA512 and SHA3 instructions	No
FEAT_SHA3		
FEAT_SM3	Cryptography support for SM instructions	No
FEAT_SM4		
FEAT_EVT	Enhanced Virtualization Traps	No
FEAT_DPB2	Cache Clean to Point of Deep Persistence	Yes
FEAT_BF16	BFloat16 extension	No
FEAT_AA32BF16	AArch32 BFloat16 extension	No
FEAT_I8MM	Int8 Matrix Multiplication	No
FEAT_AA32I8MM	AArch32 Int8 Matrix Multiplication	No
FEAT_F32MM	Single-precision Matrix Multiplication	No
FEAT_F64MM	Double-precision Matrix Multiplication	No
FEAT_PAuth	Pointer Authentication	Yes
FEAT_EPAC	Enhanced Pointer authentication	No
FEAT_PACQARMA5	Pointer authentication - QARMA5 algorithm	No
FEAT_PACIMP	Pointer authentication - IMPLEMENTATION DEFINED algorithm	No
FEAT_PACQARMA3	Pointer authentication - QARMA3 algorithm	Yes
FEAT_PAuth2	Enhancements to pointer authentication	Yes
FEAT_FPAC	Faulting on AUT* instructions	Yes
FEAT_FPACCOMBINE	Combined pointer authentication instructions	Yes
FEAT_CONSTPACFIELD	PAC algorithm enhancement	Yes
FEAT_JSCVT	JavaScript Conversion instruction	Yes, when NEON_FPm == 1
FEAT_NV	Nested Virtualization	No
FEAT_LRCPC	Weaker release consistency	Yes
FEAT_FCMA	Floating-point Complex Number support	Yes, when NEON_FPm == 1
FEAT_CCIDX	Cache extended number of sets	No
FEAT_SPEv1p1	Arm v8.3 Statistical Profiling Extensions	No
FEAT_DoPD	Arm v8.3 Debug over Power-Down	Yes
FEAT_SEL2	Secure EL2	Yes
FEAT_NV2	Enhanced support for Nested Virtualization	No
FEAT_S2FWB	Stage 2 Forced Write-Back	Yes
FEAT_DIT	Data Independent Timing	Yes
FEAT_IDST	ID Space Trap handling	Yes
FEAT_FlagM	Arm v8.4 Condition flag Manipulation	Yes
FEAT_LSE2	Arm v8.4 Large System Extensions	Yes

Feature name	Description	Implemented by the processor
FEAT_LRCPC2	Arm v8.4 enhancements to weaker release consistency	Yes
FEAT_TLBIOS	TLB maintenance and TLB range instructions	Yes
FEAT_TLBIRANGE		
FEAT_TTL	Translation Table Level	Yes, when $VMSA_m == 1$
FEAT_BBM	Change in size of page table mappings	Yes, when $VMSA_m == 1$
FEAT_CNTSC	Generic Counter Scaling	Yes
FEAT_RASv1p1	RAS extensions for Arm v8.4	Yes
FEAT_DoubleFault	Double Fault Extension	No
FEAT_Debugv8p4	Arm v8.4 Debug relaxations and extensions	Yes
FEAT_PMUv3p4	Arm v8.4 PMU extensions	Yes
FEAT_TRF	Arm v8.4 Self-hosted Trace extensions	Yes
FEAT_TTST	Small translation tables	Yes, when $VMSA_m == 1$
FEAT_FlagM2	Arm v8.5 Condition flag Manipulation	No
FEAT_FRINTTS	Floating-point to integer	No
FEAT_ExS	Context synchronization and exception handling	No
FEAT_GTG	Guest Translation Granule size	No
FEAT_BTI	Branch Target Identification	No
FEAT_EOPD	Preventing ELO access to halves of the address map	Yes, when $VMSA_m == 1$
FEAT_RNG	Random Number Generator	No
FEAT_MTE	Memory Tagging extension	No
FEAT_MTE2		
FEAT_MTE3	MTE Asymmetric Fault Handling	No
FEAT_PMUv3p5	Arm v8.5 PMU extensions	No
FEAT_RNG_TRAP	Trapping support for RNDR/RNDRRS	No
FEAT_ECV	Enhanced Counter Virtualization	No
FEAT_FGT	Fine-Grained Traps	No
FEAT_TWED	Trapping of WFE	No
FEAT_AMUv1p1	Armv8.6 AMU extensions	No
FEAT_MPAMv0p1	Arm v8.6 MPAM extension	No
FEAT_MPAMv1p1		
FEAT_MTPMU	Multi-threaded PMU extensions	No
FEAT_AFP	Alternate floating-point behavior	No
FEAT_RPRES	Increased precision of Reciprocal Estimate and Reciprocal Square Root Estimate	No
FEAT_LS64	Support for 64 byte loads/stores	No
FEAT_LS64_V		
FEAT_LS64_ACCDATA		
FEAT_WFXT	WFE and WFI instructions with timeout	No

Feature name	Description	Implemented by the processor
FEAT_HCX	Support for the HCRX_EL2 register	No
FEAT_LPA2	Larger physical address for 4KB and 16KB translation granules	No
FEAT_XS	XS attribute	No
FEAT_PMUv3p7	Armv8.7 PMU extensions	No
FEAT_SPEv1p2	Armv8.7 SPE features	No
FEAT_MOPS	Standardization of memory operations	No
FEAT_HBC	Hinted conditional branches	No
FEAT_NMI	Non-maskable Interrupts	No
FEAT_TIDCP1	ELO use of IMPLEMENTATION DEFINED functionality	No
FEAT_CMOW	Control for cache maintenance permission	No
FEAT_PMUv3p8	Armv8.8 PMU extensions	No
FEAT_HPMN0	Setting of MDCR_EL2.HPMN to zero	No
FEAT_PMUv3_TH	Event counting threshold	No
FEAT_SPEv1p3	Armv8.8 Statistical Profiling Extensions	No
FEAT_Debugv8p8	Debug v8.8	No

3.2.2 Execution state

The Arm®v8-R AArch64 architecture has only one Execution state, AArch64.

The Execution state defines the processor execution environment, including:

- Supported register widths.
- Supported instruction sets.
- Significant aspects of:
 - The execution model.
 - *Protected Memory System Architecture* (PMSA).
 - *Virtual Memory System Architecture* (VMSA).
 - The programmers' model.

3.2.3 Exception levels

In the Arm®v8-R AArch64 architecture, execution occurs at one of three Exception levels, EL0, EL1, and EL2. The Exception level determines the level of privilege where:

- EL0 has the lowest software execution privilege. Execution at EL0 is called unprivileged execution.
- Moves to a higher Exception level, such as from EL0 to EL1, indicate increased software execution privilege.

- EL2 provides support for processor virtualization.

The architecture does not specify what software runs at each Exception level, and such choices are outside the scope of the architecture. However, the following is a common usage model for the Exception levels:

EL0	Application.
EL1	Operating System.
EL2	Hypervisor.



Unlike AArch32, the AArch64 execution state does not sub-divide any of the Exception levels into different modes.

Changing Exception levels

When an exception is taken, the processor changes to the Exception level that supports the handling of the exception.

Taking an exception	An exception is <i>generated</i> when the processor first responds to an exceptional condition. The processor state at this time is the state the exception is <i>taken from</i> . The processor state immediately after taking the exception is the state the exception is <i>taken to</i> .
Returning from an exception	To return from an exception, the processor must execute an exception return instruction. The processor state when an exception return instruction is committed for execution is the state the exception <i>returns from</i> . The processor state immediately after the execution of that instruction is the state the exception <i>returns to</i> .

Execution can move between different Exception levels only on taking an exception, or on returning from an exception. Movement between Exception levels follows these rules:

- On taking an exception, the Exception level either increases or remains the same. An exception cannot be taken to a lower Exception level.
- On returning from an exception, the Exception level either decreases or remains the same. An exception cannot return to a higher Exception level.
- There is no exception handling at level EL0. Exceptions must be handled at a higher Exception level.

The Exception level that execution changes to or remains in, on taking an exception, is called the *target* Exception level of the exception, and:

- Every exception type has a target Exception level that is either:
 - Implicit in the nature of the exception.
 - Defined by configuration bits in the system registers.
- An exception cannot target EL0.

3.2.4 Instruction set architecture

The Arm®v8-R AArch64 architecture supports the A64 instruction set with Arm®v8-R AArch64 *Instruction Set Architecture* (ISA) extensions.

Arm®v8-R AArch64 ISA extensions add the following modifications to the original A64 ISA.

Table 3-2: Arm®v8-R AArch64 ISA extensions

Instructions	Change from A64
DFB	New instruction
DSB	Redefined
DMB	Redefined
DCPS3	Not supported
SMC	Not supported

The A64 instruction set provides access to 64-bit wide integer registers and data operations. Instruction opcodes, however, are still 32-bit long, not 64-bit long.

The Arm®v8-R AArch64 architecture does not support the A32 or T32 instruction sets.

3.2.5 Data types

The Arm®v8-R AArch64 architecture supports the following integer data types:

- Byte (8 bits).
- Halfword (16 bits).
- Word (32 bits).
- Doubleword (64 bits).

The Arm®v8-R AArch64 architecture also supports half-precision, single-precision, and double-precision floating-point data types as well as 64-bit and 128-bit wide vectors.

3.2.6 Arm®v8-R AArch64 registers

The Arm®v8-R AArch64 architecture has the following registers.

General-purpose registers

The Arm®v8-R AArch64 architecture provides thirty-one 64-bit general-purpose registers for instruction processing. General-purpose registers are accessible at all times and at all Exception levels. Each register can be accessed as:

- A 64-bit general-purpose register named X0 to X30.
- A 32-bit general-purpose register named W0 to W30.

The X30 general-purpose register is used as the procedure call link register.

In addition to the thirty-one general-purpose registers, there are also following special registers:

- Three 64-bit dedicated *Stack Pointer* (SP) registers for Exception levels EL0, EL1, and EL2.
- A 64-bit *Program Counter* (PC) holding the address of the current instruction. Software cannot write directly to the PC.
- Two *Exception Link Registers* (ELR) holding the exception return address for Exception levels EL1 and EL2.
- Two *Saved Program Status Registers* (SPSR) holding the state on taking exceptions for Exception levels EL1 and EL2.
- If the Cortex®-R82 processor is configured with Neon™ technology, thirty-two 128-bit Advanced SIMD and floating-point registers. These registers can be accessed as 32-bit registers S0-S31, or as 64-bit registers D0-D31, or as 128-bit registers Q0-Q31, but these are different views of the same data.

Process state, PSTATE

Process state or PSTATE is an abstraction of process state information. PSTATE holds information including:

- Flags that can be set by certain instructions and that determine the behavior of other instructions.
- Status bits that reflect the current Exception level and other states of the processor.
- Control bits that determine, for example, interrupt masking and data endianness.

System registers

System registers provide system control or status reporting. For example, a register might provide syndrome information about an abort exception that the core has taken, or provide a control to enable or disable a cache.

The System registers use a standard naming format, <register_name>.<bit_field_name>, to identify specific registers and the control and status bits within a register. Bits can also be described by their numerical position in the form <register_name>[x:y] or the generic form bits[x:y].

The System registers include:

- ID registers.
- General system control registers.
- Debug registers.
- Generic Timer registers.
- Performance Monitors Registers.
- GIC CPU interface registers.

3.2.7 Memory model

The Cortex®-R82 processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word.

The Cortex®-R82 processor can access halfwords, words, and doublewords in memory as either:

- Big-endian format.
- Little-endian format.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information about big-endian and little-endian memory systems.



Note

Some performance optimizations in the Cortex®-R82 processor memory system are only activated when memory is accessed in little-endian format. For best performance, Arm recommends using the Cortex®-R82 processor in little-endian format.



Note

Instructions are always little-endian.

3.2.7.1 Memory types

The Arm®v8 architecture provides mutually exclusive memory types. System registers in *Memory Protection Unit* (MPU) define the memory types and attributes for each region in the memory map.

The memory types are:

Normal	This is generally used for bulk memory, both read/write and read-only.
Device	This is generally used for peripherals, which might be read-sensitive or write-sensitive. The Arm architecture restricts how accesses to Device memory may be ordered, merged, or speculated.

The Arm®v8 architecture divides Device memory into several subtypes. These relate to 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 transactions from the interconnect.

The following table describes the Arm®v8 architecture memory types.

Table 3-3: Arm®v8 architecture memory types

Memory type	Comment
GRE	Similar to Normal non-cacheable, but does not permit speculative accesses.
nGRE	Treated as nGnRE inside the Cortex®-R82 processor, but can be reordered by the external interconnect.
nGnRE	Corresponds to Device in the Armv7 architecture.
nGnRE	Corresponds to Strongly Ordered in Arm® Armv7 architecture. Treated the same as nGnRE inside the Cortex®-R82 processor, but reported differently on the bus sideband signals.

3.2.7.2 Memory system architecture

The Arm®v8-R AArch64 architecture supports the following memory system architectures:

- *Protected Memory System Architecture* (PMSA) at both EL1 and EL2. This is mandatory.
- *Virtual Memory System Architecture* (VMSA) at EL1. This is optional.

The Arm®v8-R AArch64 architecture allows either of the following memory system configurations for an implementation:

- PMSA at EL1 and PMSA at EL2.
- PMSA and VMSA at EL1 and PMSA at EL2.

The hypervisor running at EL2 can select the memory system architecture for each guest OS. This enables the hypervisor to support multiple guest operating systems utilizing either PMSA or VMSA on a per guest basis.

The Arm®v8-R AArch64 architecture supports two translation regimes:

- EL1 MPU or MMU handles stage 1 translation of EL1 and EL0 translation regime.
- EL2 MPU handles stage 1 translation of EL2 translation regime and stage 2 translation of EL1 and EL0 translation regime.

If an implementation only supports PMSA at EL1, the *Virtual Address* (VA), *Intermediate Physical Address* (IPA), and *Physical Address* (PA) are all the same and translation operation reduces to memory attribute and permission checks.

3.2.8 Security model

The Arm®v8-R AArch64 architecture does not support the EL3 Exception level and therefore there is no Secure Monitor to support switching between accesses to Secure and Non-secure physical memory address space. The Arm®v8-R AArch64 architecture always operates in Secure state at all Exception levels.

The Arm®v8-R AArch64 architecture supports two Secure translation regimes that determine whether the output address is in Secure or Non-secure physical memory address space. This means

that the Cortex®-R82 processor always operates in Secure state but the Cortex®-R82 processor can access both Secure and Non-secure physical memory address space.

3.3 Advanced SIMD and floating-point

Advanced SIMD is a media and signal processing architecture.

Floating-point performs half-precision, single-precision, and double-precision floating-point operations.



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

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

4. Clocks and resets

This chapter describes the clocks and resets of the Cortex®-R82 processor.

4.1 Clocks and clock enables

The Cortex®-R82 processor requires clock signals for the cores, internal logic, and external interfaces. The Cortex®-R82 processor provides clock enables on some interfaces allowing them to operate at an integer division of the main processor clock.

The Cortex®-R82 processor is organized as a single cluster containing up to 8 cores that runs synchronously to the external memory system.

All clocks can be driven fully asynchronously to each other. The Cortex®-R82 processor contains all the necessary synchronizing logic for crossing between clock domains. There are no clock dividers and no latches in the design. The entire design uses the rising edge of the clock.

The following table describes the Cortex®-R82 processor clock input and output signals.

Table 4-1: Clock signals

Signal	Description
SCLK	The main system clock for all cluster and core logic including the memory system interfaces and the GIC interface.
PCLK	<p>The clock for the DebugBlock and the debug APB interface in the cluster.</p> <p>Note: The DebugBlock and the Cortex®-R82 processor both have PCLK inputs. You might choose to connect these to the same clock. Alternatively, you might choose to place an asynchronous bridge between the two components, in which case they might be different clocks.</p>
ATCLK	The clock for the ATB trace bus outputs from the cluster.
PERIPHCLK	The clock for Utility bus and peripheral logic inside the cluster such as timers, clock management logic, and power management logic.

The Cortex®-R82 processor provides clock enable inputs for the following interfaces to allow implementation of external logic to run at a lower synchronous frequency.

- *Main Manager* (MM) interface.
- *Main Accelerator Coherency Port* (MACP) subordinate interface.
- *Low-latency RAM* (LLRAM) manager interface.
- *ACE-Lite Subordinate* (ACELS) interface.
- *Shared Peripheral Port* (SPP) manager interface.
- *Low-latency Peripheral Port*<m> (LLPP<m>) manager interface.
- *Generic Interrupt Controller* (GIC) AXI5-Stream interface.

- Timers interface.

These interfaces can be timed as multicycle paths when they operate at an integer division of the main clock by using clock enable signals.

The following table describes the clock enable input signals and their scope.

Table 4-2: Clock enable signals

Signal	Scope
ACLKENM	MM interface
ACLKENA	MACP subordinate interface
ACLKENL	LLRAM manager interface
ACLKENS	ACELS interface
ACLKEND	SPP manager interface
ACLKENP<m>	LLPP<m> manager interface
ACLKENG	GIC AXI5-Stream interface
CNTCLKEN	Timers clock enable.

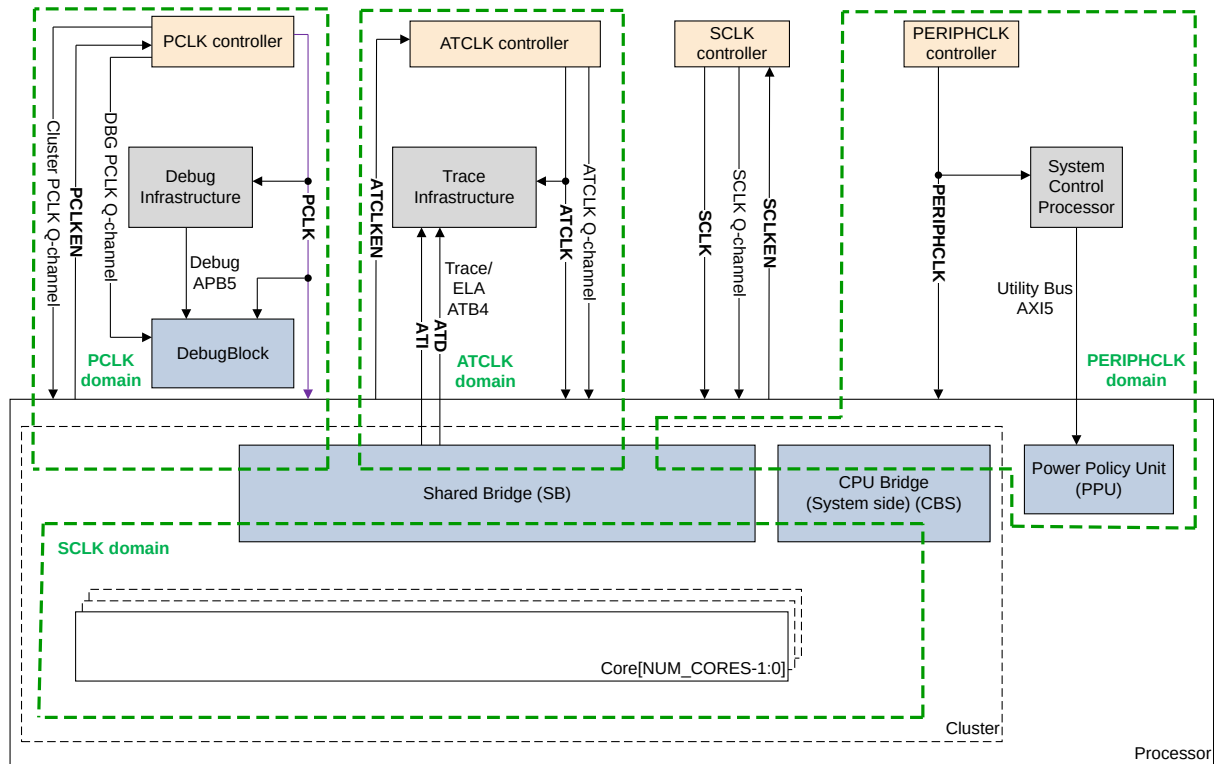
While there is no functional requirement for the clocks to have any relationship with each other, the Cortex®-R82 processor is designed with the following assumptions to achieve optimal performance and minimize latency:

- SCLK should be set to the maximum achievable frequency for the optimal system performance.
- SCLK can run at synchronous n:1 frequency with the external interconnect, avoiding the need for an asynchronous bridge between them.
- SCLK can run at synchronous n:1 frequency with the external GIC, avoiding the need for an asynchronous bridge between them.
- PCLK and ATCLK can run at the same frequency as the relevant SoC components that they connect to. This would typically be approximately 25% of the maximum SCLK frequency.
- PERIPHCLK contains the architectural timers, and software performance can be impacted if reads to these registers take too long. Therefore, Arm® recommends that PERIPHCLK is run at least 25% of the maximum SCLK frequency.

4.2 Clock domains

Most of the Cortex®-R82 processor logic, including logic within each core, operates in a single clock domain, SCLK.

The following figure shows the Cortex®-R82 processor clock domains and clock enable signals.

Figure 4-1: Clock domains and clock enable signals**Note**

The DebugBlock is shown in a common PCLK domain with the cluster debug logic. However, the DebugBlock can be placed in a different clock domain provided asynchronous bridges are inserted on the APB interfaces between the DebugBlock and the cluster.

Each clock domain shown in the figure also outputs a clock enable signal. The clock enable signal is output from the *Power Policy Unit* (PPU) and indicates if the clock is required for that clock domain or not. For example, if the power domain associated with that clock domain is OFF, then the clock can be gated and Q-Channels ignored.

The following table describes the relationship between the clock enable signals, Q-Channel and the final clock.

xCLKEN	Q-state	Final clock
0	X (do not care)	0
1	Q_STOPPED	0
1	Q_RUN	1

4.3 Resets

The Cortex®-R82 processor provides two cold (powerup) and active-LOW reset inputs and three warm (powered-on) and active-LOW reset outputs. The Cortex®-R82 processor also provides programmable resets to allow resetting of individual parts of the design through integrated *Power Policy Units* (PPUs).

The following table describes the Cortex®-R82 processor reset signals.

Table 4-4: Reset signals

Signal	Direction	Description
nRESET	Input	A global processor-wide Cold reset signal for all resettable registers in the SCLK domain excluding the DebugBlock.
nMBISTRESET	Input	A global processor-wide Cold reset signal for all resettable registers required for MBIST functionality (SCLK domain). It is intended for use by an external MBIST controller and allows it to avoid controlling the reset logic in the SoC.
nPRESET	Output	A reset signal for all resettable registers in the DebugBlock (PCLK domain).
nSRESET	Output	A reset signal for all resettable registers in the SCLK domain.
nATRESET	Output	A reset signal for all resettable registers in the ATCLK domain.

All reset inputs can be asserted (HIGH to LOW) and deasserted (LOW to HIGH) asynchronously for a minimum of 10 or more PERIPHCLK cycles. Reset synchronization logic inside the Cortex®-R82 processor ensures that reset deassertion is synchronous for all resettable registers inside those reset domains. The clock does not need to be present for reset assertion and only PERIPHCLK (for the cluster) or PCLK (for the DebugBlock nPRESET) needs to be present for reset deassertion. However, the reset might not take effect in other clock domains until the relevant clock for that domain is active.



Note

You can use the Cortex®-R82 processor reset output signals which are driven from the PPUs to reset any external logic that is in the same clock domain as the relevant parts of the cluster. For example, the nPRESET input to the DebugBlock can be connected to the nPRESET output of the cluster that is driven by the cluster PPU. This prevents possible synchronization problems. All the Cortex®-R82 processor reset outputs are generated in the PERIPHCLK domain and therefore must be synchronized before use in the destination component.

The Cortex®-R82 processor allows you to reset individual parts of the processor by programming the integrated PPUs. The following table describes the programmable resets. See [4.4 Resetting with PPUs](#) on page 55 on programming the PPU to control resets.

Table 4-5: Programmable resets

Programmable reset	Affected clock domains	Description
Core Cold reset	SCLK	Per-core Cold reset for all resettable registers in a specific core.

Programmable reset	Affected clock domains	Description
Core Warm reset	SCLK	Per-core Warm reset for all resettable registers in a specific core, except the Debug registers, ETM registers, and RAS registers.
Cluster Cold reset	SCLK, PCLK, ATCLK, PERIPHCLK	Cold reset for all resettable registers in the Cortex®-R82 processor and the DebugBlock, except the PPU.
Cluster Warm reset	SCLK	Warm reset for all resettable registers in the Cortex®-R82 processor, except the DebugBlock, the PPU, and the logic for the Utility bus, debug, ETM, and RAS functionality.

The Cortex®-R82 processor has per-core CPUHALT<m> input pins. When a core comes out of reset and is ready to start fetching from the reset vector, it checks the value of its corresponding CPUHALT<m> input pin:

- While CPUHALT<m> is HIGH, the core waits and does not start fetching from the reset vector.
- While CPUHALT<m> is LOW, the core starts fetching from the reset vector and thereafter ignores CPUHALT<m>.

You can use the CPUHALT<m> pins to enable preloading of the *Instruction Tightly Coupled Memories* (ITCMs) via the ACLES interface while the CPU is under halt, such that the cores can boot from them.

4.4 Resetting with PPUs

The *Power Policy Units* (PPUs) control the power management features of the cluster and cores using a software interface. There is one PPU for the cluster and one for each core within the Cortex®-R82 processor.

Certain power state changes, for example, powering up the cluster from a powered down state, includes implicit resets to internal logic. This internal reset is managed by the PPU controlling the transition between the two state modes and does not require an external signal to be asserted or explicit programming of the PPU. For more information on what internal reset actions result from power mode changes, see [6.7 Explicit resetting of cluster and cores and debug recovery](#) on page 88.

5. Power management

This chapter describes the power domains and the power modes in the Cortex®-R82 processor.

5.1 About power management

The Cortex®-R82 processor provides mechanisms to minimize both dynamic and static power dissipation.

The dynamic power management is achieved through local, regional, and architectural clock gating.

The static power management is achieved through dynamic retention capabilities enabled by multiple power domains and modes.

5.2 Voltage domain

The Cortex®-R82 processor has one voltage domain.

The DebugBlock typically resides in the same voltage domain as the Cortex®-R82 processor. However, you can place the DebugBlock in a separate voltage domain if necessary. In this case, the implementer has to place appropriate bridges on the APB interfaces between the DebugBlock and the Cortex®-R82 processor.

5.3 Clock gating

The Cortex®-R82 processor includes extensive clock gating to reduce dynamic power consumption.

Clock gating includes:

- Local clock gating inferred by the synthesis tools.
- Regional clock gating with instantiated clock gates covering a larger region of logic. As a subset of this, the Cortex®-R82 processor implements architectural clock gating, defined as clock gating the majority of a core's logic when that core is in a low-power state, such as the *Wait for Interrupt* (WFI) state.
- Hierarchical clock gating which is performed externally to the Cortex®-R82 processor when all components on a clock domain are idle.

5.3.1 Local and regional clock gating

The local and regional clock gating is performed automatically and needs no external support.

The Cortex®-R82 processor has been designed to enable synthesis tools to automatically infer local clock gates for groups of flip-flops. These clock gates disable the clock and therefore reduce the dynamic power that is consumed by the flip-flops and logic local to the clock gate.

Regional clock gating in the Cortex®-R82 processor allows the clock for larger regions of logic to be disabled when idle, which further reduces dynamic power consumption. The Cortex®-R82 processor contains regional clock gates for all components within the cluster. The Cortex®-R82 processor automatically enables and disables the regional clock gates according to its functional requirements.

The Cortex®-R82 processor also implements architectural clock gating for the clock to a core when that core is in a low-power state, such as when executing a *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) instruction. The architectural clock gates for each core are implemented within the CPU Bridge unit. The Cortex®-R82 processor automatically enables and disables the architectural clock gates according to its functional requirements.

5.3.2 Hierarchical clock gating

The Cortex®-R82 processor allows further power savings by supporting the clock to be gated off higher up in the clock tree. To do this, the Cortex®-R82 processor provides Q-Channels for SCLK, ATCLK, and PCLK clock domains that can be used by an external clock controller to gate the clock when all components in that clock domain are idle.



There are two Q-Channels to hierarchically gate PCLK components, one for the cluster logic and one for the DebugBlock. This is because the DebugBlock can be implemented in a separate power domain and can be independently powered off and on.

The exception to this is the PERIPHCLK, because the PERIPHCLK is expected to drive the logic that is always ON and that is responsive to the events such as timers.

The Cortex®-R82 processor has several interfaces that connect to other components in the SoC that are likely to be in the same clock domains. For example, the Cortex®-R82 processor *Generic Interrupt Controller* (GIC) CPU interface signals might be in the same clock domain as the external GIC distributor, and the Cortex®-R82 processor ATB interface signals might be in the same clock domain as the external trace infrastructure. Hierarchical clock gating of these components when they are idle allows further power reduction.

Most of the Cortex®-R82 processor, including the logic within each core, operates in a single clock domain, SCLK. When all cores and all cluster components are inactive, a Q-Channel allows an external clock controller to gate the whole SCLK domain.

The hierarchical clock gates are implemented within the *Shared Bridge* (SB) unit.

5.3.2.1 Wait for Interrupt and Wait for Event

Wait for Interrupt (WFI) and *Wait for Event* (WFE) are architectural features that are used to put each core in the Cortex®-R82 processor in a low-power Standby mode by regional disabling the clock at the top of the clock tree.

To reduce dynamic power, each core in the Cortex®-R82 processor can request entry into a low-power state using the `WFI` and `WFE` instructions. In the low-power state, most of the clocks in a core are disabled while keeping the core powered up. This reduces the power drawn to the static leakage current, leaving a small clock power overhead to enable the core to wake up.

In addition to the per-core `WFI` and `WFE` low-power states, the clock to (almost all) the L2 and *LLRAM Coherency Unit* (LCU) logic is automatically disabled when the cluster is sufficiently idle.

A `WFI` or `WFE` instruction completes when:

- All outstanding load instructions are completed.
- All store instructions are completed.
- All bus traffic is completed.

While a core is in the low-power state, the clocks in the core are temporarily enabled under the following conditions:

- A snoop request from the L2 cache that must be serviced by the L1 data cache.
- An APB access to the debug, trace, or core *Performance Monitoring Unit* (PMU) registers residing in the core power domain.
- An access request from the *Generic Interrupt Controller* (GIC) distributor to the GIC CPU interface.
- An access from the Utility bus to *Reliability, Availability, and Serviceability* (RAS) registers.

While the clocks in the core are temporarily enabled, the core remains in the `WFI` or `WFE` low-power state.

WFE wake up event signaling

- A *Send Event* (`SEV`) instruction signals a `WFE` wake up event to other clusters by asserting the `EVENTOREQ` output.
- The `EVENTIREQ` input indicates that another cluster or system component has signaled a `WFE` wake up event.

System global exclusive monitor signaling

Any global exclusive monitor in the system must be able to generate an event when it is cleared. This event must be signaled to the cluster using the `EVENTIREQ` input.

5.4 Power domains

The Cortex®-R82 processor supports several different power domains. However, you do not need to implement all the available power domains. The implementation choices, such as the number of cores or L2 cache implementation, determine the number and type of power domains that are able to be implemented.

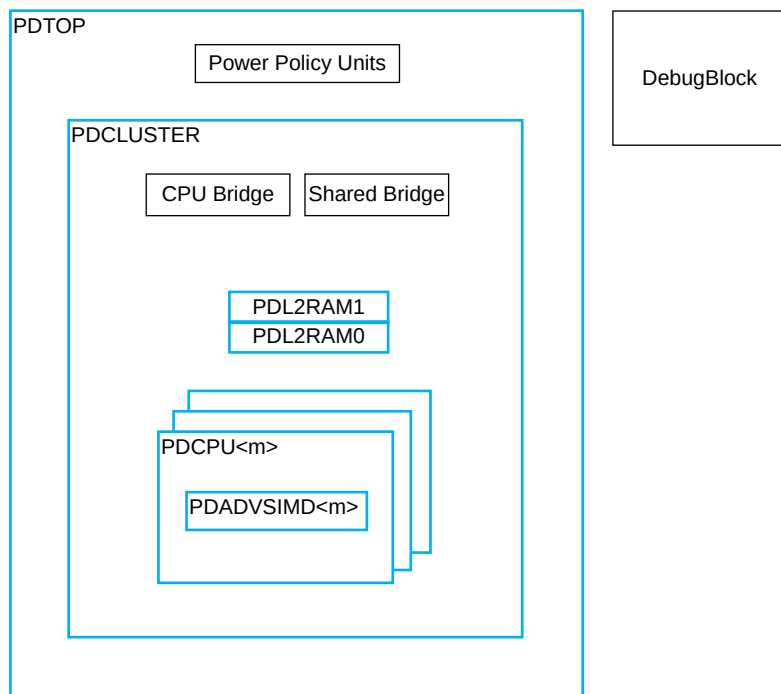
The Cortex®-R82 processor supports PDTOP, PDCLUSTER, PDL2RAM0, PDL2RAM1, PDCPU<m>, and PDADVSIMD<m> power domains.



PDADVSIMD<m> is present if the core <m> is configured to support Advanced SIMD and floating-point (NEON_FP<m> = 1).

The following figure shows the supported power domains. Each box with blue boundaries indicates a separate power domain.

Figure 5-1: Processor power domains



PDTOP

The Cortex®-R82 processor has a top-level power domain, PDTOP, that allows the whole processor to be turned off. PDTOP is expected to be in the same power domain as other SoC

components. The only cluster logic in PDTOP power domain is the *Power Policy Units* (PPUs) which will typically be always On. This is because the PPUs need to power down the other power domains including PDCLUSTER while remaining active.

The PDTOP power domain must be powered up before any of the other power domains are powered up. It must only be powered down after the other power domains have been powered down.

The DebugBlock is designed to be included in a separate Debug power domain with other Debug components.

PDCLUSTER

The whole cluster, excluding the PPUs, belongs to a single power domain, PDCLUSTER. This allows the Cortex®-R82 processor to be put in a lower power state while the PPUs stay operational to handle power transitions.

PDL2RAM0 and PDL2RAM1

PDL2RAM0 and PDL2RAM1 power domains enable the independent power control of the L2 cache RAMs. PDL2RAM0 and PDL2RAM1 power domains allow:

- The L2 cache to support operation with only half of the RAMs active, when multiple cores are turned off or in retention.



PDL2RAM0 and PDL2RAM1 can be controlled individually only if `L2_SLICES` is set to 2.

- The Cortex®-R82 processor to turn off or put in retention all the L2 cache RAMs.

PDGPU<m>

Each core within the Cortex®-R82 processor has its own power domain, PDGPU<m>, to allow the cores to be powered down individually.

PDADVSIMD<m>

The Advanced SIMD and floating-point block in each core within the Cortex®-R82 processor is also part of the power domain for that core. However, to support independent retention control, each Advanced SIMD and floating-point block also has its own power domain, PDADVSIMD<m>, for isolating it from the surrounding domain. This allows the Advanced SIMD and floating-point block to be put in retention while the rest of the core is active.

Clamping and isolation cells between power domains are inferred from the supplied UPF files rather than instantiated in the RTL.

The following table shows the power domains that the Cortex®-R82 processor supports.

Table 5-1: Power domain description

Power domain	Description
PDTOP	This domain contains all cluster logic including the <i>Power Policy Units</i> (PPUs).
PDCLUSTER	This domain contains the <i>Shared Bridge</i> (SB), <i>CPU Bridge</i> (CB) (both system side and CPU side CBs), trace and debug routing infrastructure, <i>LLRAM Coherency Unit</i> (LCU), <i>Shared AXI Subordinate Unit</i> (SAXIS), L2 coherency logic, and L2 cache RAMs. It excludes the PPUs.
PDL2RAM0	This domain contains the first half of the L2 cache RAMs.
PDL2RAM1	This domain contains the second half of the L2 cache RAMs.
PDCPU<m>	This domain contains all core logic including the optional Advanced SIMD and floating-point block, the <i>Tightly Coupled Memories</i> (TCMs) and L1 cache RAMs, and Debug registers that are associated with the core <m> where m is the core number in the range of 0-7. If a core is not present, the corresponding power domain is not present.
PDADVSIMD<m>	This is a power domain for Advanced SIMD and floating-point block in core <m> to implement functional retention. <m> is the core number in the range of 0-7. If the Advanced SIMD and floating-point block is not present in core <m>, the PDADVSIMD<m> power domain is not present.

5.5 Power mode control

Power mode control is distributed between power management software, the cluster, and the *Power Policy Units* (PPUs) integrated within the cluster.

A component in the SoC such as a *System Control Processor* (SCP) can program the PPUs over the Utility bus to set the appropriate power policy. If your system does not have an SCP component, the Utility Bus can be connected to one of the Cortex®-R82 processor ports or to your SoC interconnect, so one of the cores in the Cortex®-R82 processor can program the PPUs.

If your system does not need to perform any power transitions, the Cortex®-R82 processor can also be configured so that the PPUs are powered on at reset and never require any programming. See [6.9 Implications of not having a System Control Processor](#) on page 93 for more information on Cold reset state for the PPUs.

The PPUs control the low-level details of powering up, powering down, or resetting domains as necessary to implement the requested policy. The hardware performs any actions necessary to reach the requested power mode, such as gating clocks, flushing caches, or disabling coherency.

The power mode of each core can be changed independently of other cores in the cluster, however the cluster power mode is linked to the state of the cores. There is no requirement on the order that cores are powered on or off.

The PPUs and cluster logic perform all the logical functions needed to enter or exit a power mode. However, there are some steps related to the physical state such as controlling power switches or retention states. The logic to carry out these steps is not included in the Cortex®-R82 processor, because it varies depending on the technology process, library, and internal design rules. Therefore the implementer must provide a *Power Control State Machine* (PCSM) that sequences these implementation-specific steps.

The PPU interfaces to the PCSM via a P-Channel interface, where the PPU can initiate a request to a new power mode. The PCSM accepts that request when it has completed all of its required actions.

Software sets the operating requirements by writing to the following System registers:

Cluster Power Control Register (IMP_CLUSTERPWRCTLR_EL1)

To request partial L2 cache powerup or powerdown and to enable RAM retention capabilities.

Cluster Powerdown Register (IMP_CLUSTERPWRDN_EL1)

To request the power mode that the cluster is to enter after all cores have powered off. For example, memory retention mode.

CPU Power Control Register (IMP_CPUPWRCTLR_EL1)

To request core powerdown and to enable Advanced SIMD and floating-point retention capabilities.

5.6 Debug over powerdown

The Cortex®-R82 processor supports debug over powerdown which allows a debugger to retain its connection with the Cortex®-R82 processor even when the Cortex®-R82 processor is powered down. This behavior enables debug to continue through powerdown scenarios rather than having to re-establish a connection each time the Cortex®-R82 processor is powered up.

The debug over powerdown logic is part of the DebugBlock which is external to the Cortex®-R82 processor. The DebugBlock is provided as a separate component to allow implementation in a separate power domain from the processor. Having a separate debug power domain allows the connection to a debugger be maintained while the cores and cluster are powered down.

5.7 Core power modes and transitions

Each core within the Cortex®-R82 processor has a defined set of power modes and permitted transitions between these modes. The power mode of each core can be independent of other cores in the Cortex®-R82 processor.

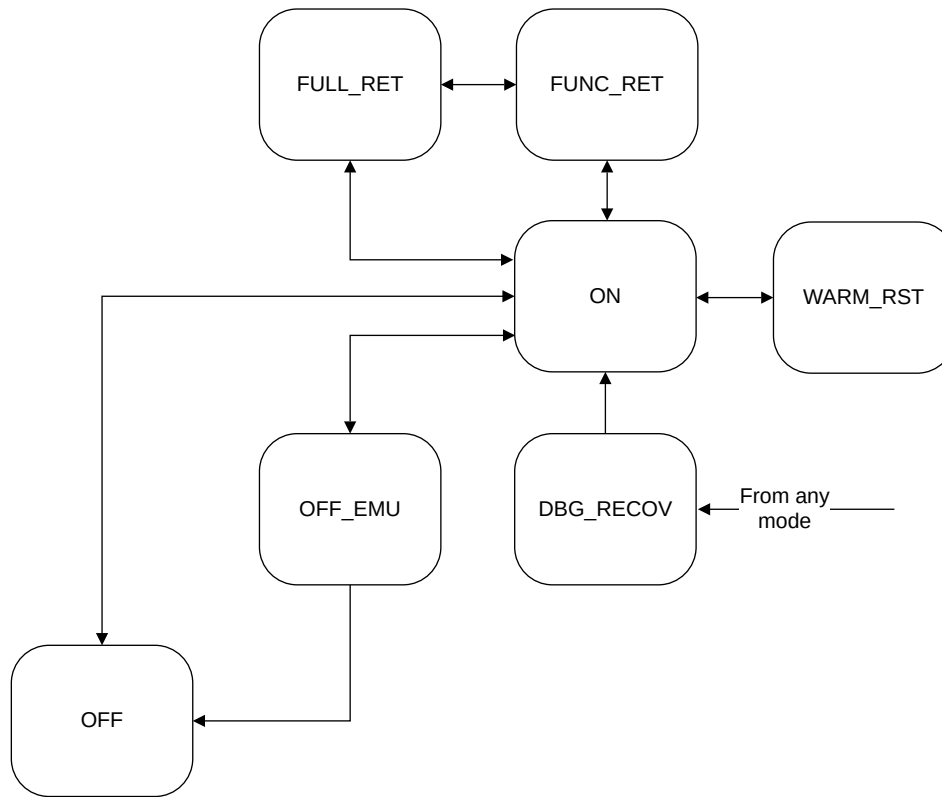
The following table shows the supported core power modes.

Table 5-2: Core power modes

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

Power mode	Short name	Description
Functional retention	FUNC_RET	<p>The core is fully powered and operational, but the Advanced SIMD and floating-point logic is in retention.</p> <p>All instructions except for Advanced SIMD and floating-point ones can execute normally. When an Advanced SIMD or floating-point instruction is encountered, the pipeline stalls until the core can transition to ON to execute the instruction.</p>
Full retention	FULL_RET	<p>The core and all the RAMs are in retention.</p> <p>In this mode, only power that is required to retain register and RAM state is available.</p> <p>The core must be in <i>Wait for Interrupt</i> (WFI) or <i>Wait for Event</i> (WFE) low-power state before it enters this mode.</p>
Off	OFF	The core is powered down.
Emulated off	OFF_EMU	<p>Emulated off mode permits you to debug the powerup and powerdown cycle without changing the software.</p> <p>In this mode, the core powerdown is normal, except:</p> <ul style="list-style-type: none"> • The clock is not gated and power is not removed when the core is powered down. • Only the Warm reset is asserted. The debug logic is preserved in the core and remains accessible by the debugger. <p>Note: Emulated off mode operation for the shared logic is identical to the operation for a core.</p>
Debug recovery reset	DBG_RECOV	<p>Debug recovery reset is used for applying a reset to the core, while preserving memory and optionally <i>Reliability, Availability, and Serviceability</i> (RAS), Debug, and Trace registers for debug purposes. The L1 cache state is preserved when transitioning from DBG_RECOV mode to ON mode. Debug recovery mode is typically used in debugging a watchdog timeout.</p> <p>Caution: This mode must not be used during normal system operation.</p>
Warm reset	WARM_RST	<p>Warm reset is a best effort to recover from system level issues while keeping the state for the trace logic and the Debug and RAS registers.</p> <p>Caution: This mode must not be used during normal system operation.</p>

The following diagram shows the supported power modes for each core and the permitted transitions between them.

Figure 5-2: Core power mode transitions

ON

In this mode, the core is on and fully operational.

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

FUNC_RET

In this mode, the Advanced SIMD and floating-point logic is in retention (inoperable but with state retained) and the remainder of the core logic is operational.

This means that if an Advanced SIMD and floating-point instruction is executed while in this mode, it is stalled until the core enters the ON mode.

When the Advanced SIMD and floating-point logic is in retention, the clock to the logic is automatically gated outside of the retained domain.

You can control the FUNC_RET by setting the IMP_CPUPWRCTLR_EL1.FPURET register bits.

FULL_RET

In this mode, all core logic and RAMs are in retention, that is the domain is inoperable but with core state retained.

This mode is typically used when the core is in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) state for an extended period of time.

When the core is in FULL_RET, there is support for Snoop and debug access so the core transitions to the ON state to process the access. The core, then, transitions back to FULL_RET without the core leaving WFI or WFE state.

The core dynamic retention can be enabled and disabled separately for WFI and WFE by software running on the core. You can program separate timeout values for entry into this mode from WFI and WFE mode:

- Use the IMP_CPUPWRCTLR_EL1.WFIRET register bits to program timeout values for entry into core FULL_RET mode from WFI mode.
- Use the IMP_CPUPWRCTLR_EL1.WFERET register bits to program timeout values for entry into core FULL_RET mode from WFE mode.

When the Cortex®-R82 processor transitions a core from FULL_RET to ON powermode:

- The L1 cache, L2 duplicate L1 tag RAM, and LCU duplicate L1 tag RAM are not invalidated.

OFF

In this mode, all core logic and RAMs are unused and can be powered down. The domain is inoperable and all core state is lost.

When the core is in OFF, any attempted debug access returns an error response on the internal debug interface indicating the core is not available.

The core can enter the OFF mode by setting the IMP_CPUPWRCTLR_EL1.PWRDN register bit.

OFF_EMU

In this mode, all core logic and RAMs are kept physically powered on. However, core Warm reset is asserted externally to emulate OFF scenario while keeping core debug state and allowing debug access.

All Debug registers retain their state and are accessible from the external debug interface. All other functional interfaces behave as if the core were OFF.

DBG_RECOV

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

In DBG_RECOV mode, the core logic including the L1 cache RAMs is powered up.

All powered-on cores and the cluster need to be put into DBG_RECOV mode. When this happens, the processor applies either a Warm reset or a Cold reset, depending on the PPU_PTCR.DBG_RECOV_PORST_EN value. Following this, the powered-on cores and cluster

should be put into the ON power mode. See [6.7 Explicit resetting of cluster and cores and debug recovery](#) on page 88 for more information.

Normally, the Cortex®-R82 processor performs the following when there is a transition to ON power mode:

- Invalidates the L1 cache, L2 duplicate L1 tag RAM, and *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAM (when there is a core transition from OFF, OFF_EMU, or WARM_RST to ON mode and a Warm reset or a Cold reset applies).
- Resets the register file and System registers which have an **UNKNOWN** reset value (when there is a core transition from OFF, OFF_EMU, or WARM_RST to ON mode and a Warm reset or a Cold reset applies).
- Resets the Debug, Trace and *Reliability, Availability, and Serviceability* (RAS) state (when there is a transition from OFF to ON mode and a Cold reset applies).

In contrast, when the Cortex®-R82 processor transitions a core from DBG_RECOV to ON power mode:

- The L1 cache, L2 duplicate L1 tag RAM, and LCU duplicate L1 tag RAM are not invalidated.
- If PPU_PTCR.DBG_RECOV_PORST_EN = 1, the Debug, Trace and RAS state is reset. If PPU_PTCR.DBG_RECOV_PORST_EN = 0, the Debug, Trace and RAS state is preserved.

Debug recovery mode can be entered from any other mode. The core *Power Policy Unit* (PPU) controls entry into this mode.



- The system must be able to program the PPUs over the Utility bus to use Debug recovery. For example, the system may have a *System Control Processor* (SCP), or the debug subsystem may connect to the Utility bus.
- Debug recovery mode 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. To go back to functional mode, the system has to go through a full Cold reset.
- Debug recovery mode can occur at any time with no guarantee of the state of the core. A P-Channel request of this type is accepted immediately, therefore its effects on the core, cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. In particular, if there were outstanding memory system transactions or shared System registers being accessed at the time of the reset, then these transactions might complete after the reset when the core is not expecting them and cause a system deadlock.
- No debug access to the core should be made in this power mode. Debug access should only be made in the ON or OFF_EMU power mode.
- If the system sends a snoop to the cluster during debug recovery mode, then depending on the cluster state, the snoop might get a response and disturb the contents of the caches, or it might not get a response and cause a system deadlock.

WARM_RST

A Warm reset resets all state except for the trace logic and the debug and RAS registers.

A Warm reset invalidates the caches and snoop filters and resets the register file and System registers with **UNKNOWN** reset values.



Caution

- If any core is put into Warm reset mode, then the cluster must also be put into Warm reset mode and the other cores must go into Warm reset mode, Off mode, or Emulated off mode. Therefore, using the core Warm reset mode has the end result of resetting the cores and the shared logic.
- The system must be able to program the PPU over the Utility bus to use Warm reset. For example, the system may use an SCP.
- Warm reset is a best effort to recover from system level issues while keeping debug and RAS registers and should not be used for functional purposes.
- No debug access to the core should be made in this power mode. Debug access should only be made in the ON or OFF_EMU power mode.
- Warm reset state can occur at any time for an ON core with no further guarantees of its state. A request of this type is accepted immediately by an ON core, therefore its effects on the core, cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. In particular, if there were outstanding memory system transactions or shared System registers being accessed at the time of the reset, then these transactions might complete after the reset when the core is not expecting them and cause a system deadlock.

5.8 Core powerdown

You must follow a specific powerdown sequence to trigger core powerdown.

To trigger core powerdown:

1. Save all architectural state.
2. Configure the *Generic Interrupt Controller* (GIC) distributor to disable or reroute interrupts away from the core.



Note

GIC distributor can be either of the following:

- An Arm GIC distributor, such as GIC-625, that connects to the GIC Stream processor ports.
 - Any other interrupt controller that might be driving the nFIQ, nIRQ, nVFIQ, or nVIRQ processor inputs.
3. Set the IMP_CPUPWRCTLR_EL1.PWRDN bit to 1 to indicate to the power controller that a powerdown is requested.

4. Execute an *Instruction Synchronization Barrier* (ISB) instruction.
5. Execute a `WFI` instruction.

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

- Disables and flushes the L1 caches.
- Removes the core from coherency.

When the `IMP_CPUPWRCTLR_EL1.PWRDN` bit is set, executing a `WFI` instruction automatically masks all interrupts and wakeup events in the core. As a result, applying reset is the only way to wake up the core from the `WFI`.

No software steps are required to bring a core into coherence after reset.

5.9 Cluster power modes

The Cortex®-R82 processor supports various cluster level low-power modes and includes hardware to handle power mode transitions with minimal software support.

The following table shows the supported power modes for the shared logic in the Cortex®-R82 processor.

Table 5-3: Cortex®-R82 processor shared logic power modes

Power mode	Short name	Description
On	ON	On mode is the normal mode of operation where all shared logic functionality is available.
Memory retention	MEM_RET	In Memory retention mode, only the L2 cache RAMs are placed in retention. The rest of the cluster including the L2 logic and the cores are powered down.
Emulated memory retention	MEM_RET_EMU	In Emulated memory retention mode, the cluster behaves logically as if it were in the MEM_RET mode, except that the shared RAMs and the cluster logic remain powered. The debug state is retained and is accessible.
Off	OFF	In Off mode, power is removed from the cluster logic and all the RAMs. Only the <i>Power Policy Units</i> (PPUs) remain powered.
Emulated off	OFF_EMU	In Emulated off mode, the cluster behaves logically as if it were in the OFF mode, except that the logic remains powered. The debug state is retained and accessible.
Debug recovery reset	DBG_RECOV	<p>Debug recovery reset is used for applying a reset to the cluster, while preserving memory and optionally <i>Reliability, Availability, and Serviceability</i> (RAS), Debug, and Trace registers for debug purposes. The L2 cache state is preserved when transitioning from DBG_RECOV mode to ON mode. Debug recovery mode is typically used in debugging a watchdog timeout.</p> <p>Caution: This mode must not be used during normal system operation.</p>

Power mode	Short name	Description
Warm reset	WARM_RST	<p>The Warm reset mode provides a Warm reset to all the shared cluster logic apart from the <i>Power Policy Units</i> (PPUs).</p> <p>Warm reset is a best effort to recover from system level issues while keeping the state for the trace logic and the Debug and RAS registers.</p> <p>Caution: This mode must not be used during normal system operation.</p>

ON

In this mode, the cluster is on and fully operational.

When a transition to the ON mode completes, the L2 cache is accessible and coherent without needing any configuration from software other than the normal architectural steps to enable caches.

MEM_RET

In Memory retention mode, the L2 cache RAMs are placed in retention while the shared logic and the cores are powered down.

It is quicker for the cluster to enter and exit MEM_RET mode as compared with going from OFF to ON mode or ON to OFF mode, for example in powerup or powerdown. This is because, the L2 cache RAMs do not need to be cleaned and the data later reloaded. Placing the L2 cache RAMs in retention also saves on energy required to write dirty data back to main memory.



Caution

The Cortex®-R82 processor remains in coherence when in MEM_RET mode. Therefore, when using this mode, beware that if other external coherent agents are active, it takes considerable time for them to access to the L2 cache RAMs. Although it is possible for external agents in the system to access the L2 cache RAMs while in retention, it comes at considerable time cost because the Cortex®-R82 processor needs to be temporarily powered up to service the access.

When the Cortex®-R82 processor transitions the cluster from MEM_RET to ONmode:

- The L2 cache, L2 duplicate L1 tag RAMs, and LCU duplicate L1 tag RAMs are not invalidated.

MEM_RET_EMU

In Emulated memory retention mode, the cluster behaves logically as if it were in the MEM_RET mode except that the cluster shared logic remains powered. This means the L2 cache RAMs, L2 duplicate L1 tag RAMs, *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs, and the rest of the cluster logic remains powered. Therefore, debug accesses can be made.

OFF

In the OFF mode, all the shared cluster logic including the L2 duplicate L1 tag RAMs, LCU duplicate L1 tag RAMs, L2 cache RAMs, and the cores are powered down. The PDCLUSTER domain is inoperable and all state is lost.

In the OFF mode, power is removed from PDCLUSTER power domain but the PDTOP power domain is still powered up including all the *Power Policy Units* (PPUs).

The Cortex®-R82 processor can be initialized into this mode on a Cold reset.

OFF_EMU

In this mode, the cluster behaves logically as if it were in the OFF mode. However, the cluster shared logic remains powered including the L2 cache RAMs, L2 duplicate L1 tag RAMs, and LCU duplicate L1 tag RAMs. Therefore, debug accesses to the cluster can still be made.

In this mode, the cluster behaves as if it were powered off for functional logic, but it allows the cluster to maintain debug access. On entering this mode, a Warm reset is applied to the cluster, resetting the functional logic but not resetting the debug logic. From the perspective of software running on the core, the cluster appears to be powered off.

DBG_RECOV

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

In DBG_RECOV mode, all the Cortex®-R82 processor shared logic including the L2 cache RAMs is powered up.

All powered-on cores and the cluster need to be put into DBG_RECOV mode. When this happens, the processor applies either a Warm reset or a Cold reset, depending on the CLUSTERPPU_PTCR.DBG_RECOV_PORST_EN value. Following this, the powered-on cores and cluster should be put into the ON power mode. See [6.7 Explicit resetting of cluster and cores and debug recovery](#) on page 88 for more information.

Normally, the Cortex®-R82 processor performs the following when there is a transition to ON power mode:

- Invalidates the L2 cache, L2 duplicate L1 tag RAMs, and LCU duplicate L1 tag RAMs (when there is a cluster power mode transition from OFF, OFF_EMU, or WARM_RST to ON mode and a Warm reset or a Cold reset applies)
- Resets the Debug, Trace and *Reliability, Availability, and Serviceability* (RAS) state (when there is a cluster power mode transition from OFF to ON mode and a Cold reset applies).

In contrast, when the Cortex®-R82 processor transitions the cluster from DBG_RECOV to ON mode:

- The L2 cache, L2 duplicate L1 tag RAMs, and LCU duplicate L1 tag RAMs are not invalidated.

- If CLUSTERPPU_PTCR.DBG_RECOV_PORST_EN = 1, the Debug, Trace and RAS state is reset. If CLUSTERPPU_PTCR.DBG_RECOV_PORST_EN = 0, the Debug, Trace and RAS state is preserved.

Debug recovery mode can be entered from any other mode. The cluster *Power Policy Unit* (PPU) controls entry into this mode.



Caution

- The system must be able to program the PPU's over the Utility bus to use Debug recovery. For example, the system may have a System Control Processor (SCP), or the debug subsystem may connect to the Utility bus.
- Debug recovery mode is strictly for debug purposes. It must not be used for functional purposes because correct operation of the cluster is not guaranteed when entering this mode. To go back to functional mode, the system has to go through a full Cold reset.
- No debug access to the cluster should be made in this power mode. Debug access should only be made in the ON or OFF_EMU power mode.
- Debug recovery mode can occur at any time with no guarantee of the state of the cluster. A request of this type is accepted immediately, therefore its effects on the core, cluster, or the wider system are unpredictable and a wider system reset might be required. In particular, if there were outstanding memory system transactions or shared System registers being accessed at the time of the reset, then these transactions might complete after the reset when the cluster is not expecting them and cause a system deadlock.

If the system sends a snoop to the cluster during this mode, then depending on the cluster state:

- The snoop might get a response and disturb the contents of the caches.
- The snoop might not get a response and cause a system deadlock.
- In the following cases, it might not be possible to enter DBG_RECOV without a Cold reset of the cluster:
 - When the cluster is in middle of a power transition which cannot complete because of the system hanging or trying to debug.
 - When the cluster is in the middle of a clock gating transition on the SCLK Q-Channel which cannot complete because of the system hanging or trying to debug.
 - The cluster is in Warm reset.
- You must choose the correct operating mode corresponding to the L2 cache partitions and L2 cache slices that were in use before Debug recovery mode.

WARM_RST

A Warm reset resets all state except for the trace logic and the debug and RAS registers.

A Warm reset invalidates the caches and snoop filters and resets the register file and System registers with **UNKNOWN** reset values.



- If any core is put into Warm reset mode, then the cluster must also put into Warm reset mode and the other cores must go into Warm reset mode, Off mode, or Emulated off mode. Therefore, using cluster Warm reset mode has the end result of resetting the cores and the shared logic.
- The system must be able to program the PPU over the Utility bus to use Warm reset. For example, the system may use an SCP.
- Warm reset is a best effort to recover from system level issues while keeping debug and RAS registers and should not be used for functional purposes.
- No debug access to the cluster should be made in this power mode. Debug access should only be made in the ON or OFF_EMU power mode.
- Warm reset mode is only expected to be used for resets triggered by a system level issue, such as a watchdog timeout, and therefore a Cold system reset might be required. In particular, if there were outstanding memory system transactions or shared System registers being accessed at the time of the reset, then these transactions might complete after the reset when the cluster is not expecting them and cause a system deadlock.
- The warm reset power mode can occur at any time with no further guarantees for the state of the cluster. A request of this type is accepted immediately by the cluster, therefore its effects on the core, cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. In particular, if there were outstanding memory system transactions or shared System registers being accessed at the time of the reset, then these transactions might complete after the reset when the core is not expecting them and cause a system deadlock.

5.10 Cluster operating modes

An operating mode is a component-specific configuration of the power modes. For the Cortex®-R82 processor, the operating modes differ in the amount of L2 cache RAM that is active.

The cluster *Power Policy Unit* (PPU) provides programming access to control the operating modes and the power modes.

The Cortex®-R82 processor supports three operating modes.

The following table shows the operating modes for the L2 cache RAMs.

Table 5-4: Operating modes for L2 cache RAMs

Operating mode	Short name	Description
L2 duplicate L1 tag RAM and LLRAM Coherency Unit (LCU) duplicate L1 tag RAM only	DL1ONLY	The L2 cache partition in each cache slice is powered down

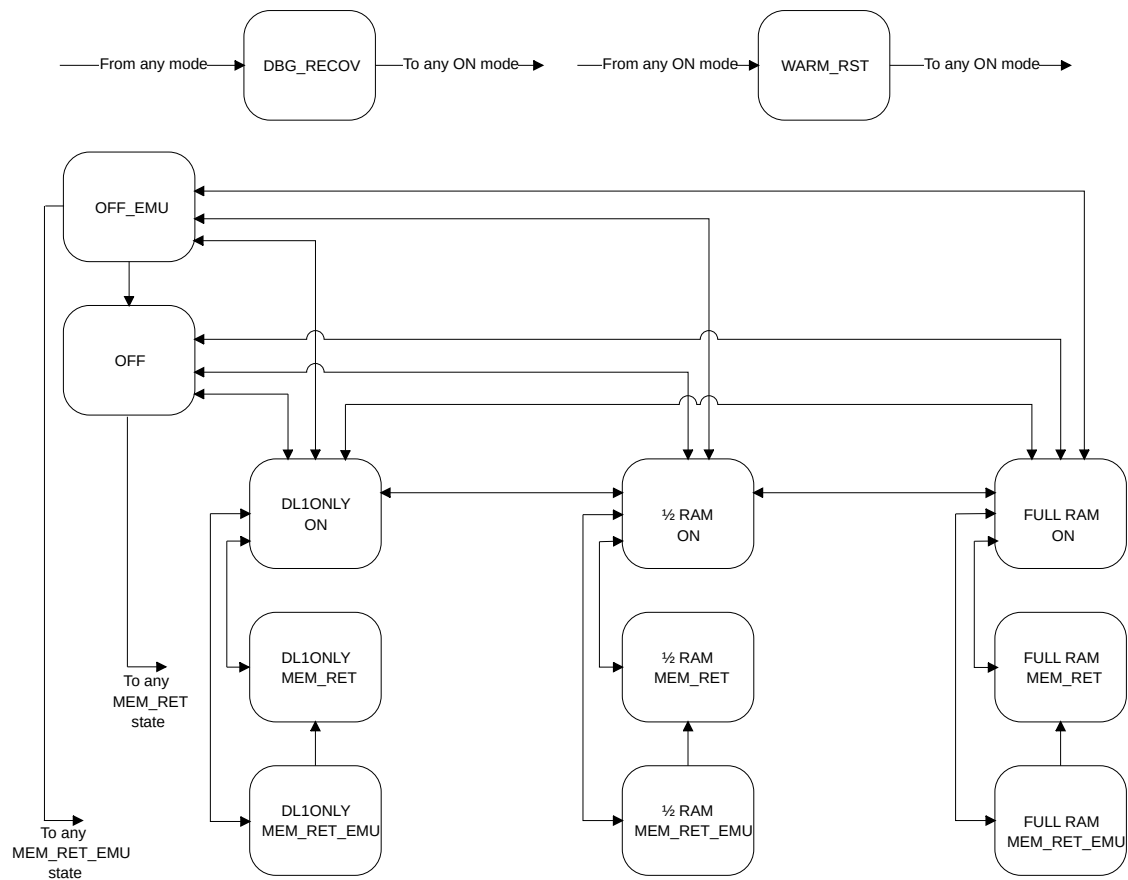
Operating mode	Short name	Description
Half L2 cache	½ RAM	One half of the L2 cache partition in each active slice is powered up Note: This mode is possible only if L2_SLICES is set to 2.
Full L2 cache	FULL RAM	All of the L2 cache partition in each active slice is powered up

5.11 Cluster PPU mode transitions

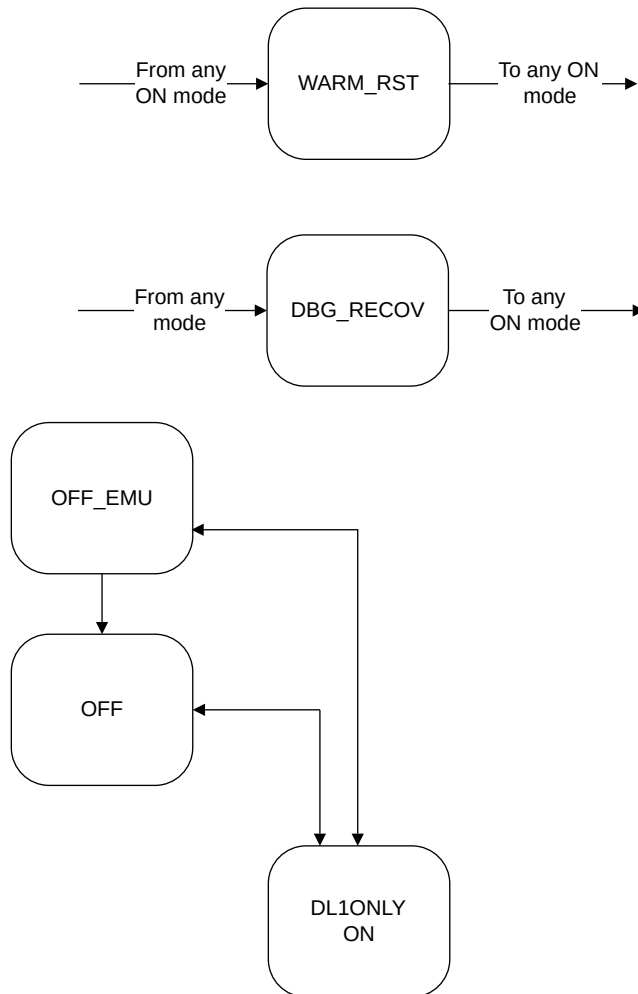
The Cortex®-R82 processor supports transitions between cluster power modes and cluster operating modes. Each combination of cluster power mode with an L2 cache RAM operating mode forms a *Power Policy Unit* (PPU) mode, for example, FULL RAM ON. Individual power modes such as ON, OFF, and OFF_EMU are also considered to be PPU modes.

The cluster PPU controls transitions between the PPU modes. Therefore, a *System Control Processor* (SCP) or any core within the Cortex®-R82 processor through a Utility bus loopback connection can program up the PPU to go to any PPU mode and the PPU would automatically schedule the necessary transitions to achieve that PPU mode.

The following figure shows the supported PPU mode transitions for the Cortex®-R82 processor.

Figure 5-3: Cortex®-R82 cluster PPU mode transitions

The following figure shows the supported PPU mode transitions for the Cortex®-R82 processor where the L2 cache is not implemented.

Figure 5-4: Cortex®-R82 cluster PPU mode transitions, no L2**FULL RAM ON**

In this PPU mode, all the shared logic including the L2 cache RAMs, L2 duplicate L1 tag RAMs, *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs, and PPUs is powered up and fully operational. When a transition to the On mode is completed the L2 cache, L2 duplicate L1 tag RAMs, and LCU duplicate L1 tag RAMs are accessible and coherent without requiring any software configuration.

1/2 RAM ON

In this PPU mode, the Cortex®-R82 processor shared logic, L2 duplicate L1 tag RAMs, and *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs are powered up but half of the L2 cache RAMs remain powered down.

DL1ONLY ON

In this PPU mode, the Cortex®-R82 processor shared logic, L2 duplicate L1 tag RAMs, and LCU duplicate L1 tag RAMs are powered up but the L2 cache RAMs remain powered down.

FULL RAM MEM_RET, 1/2 RAM MEM_RET

In this PPU mode, the L2 cache RAMs are in retention, but the rest of the Cortex®-R82 processor shared logic is powered down, apart from the PPUs. This is also known as Dormant mode. The L2 cache still contains data and if another agent in the system needs to snoop the cluster to access that data then the cluster needs to transition to an On mode before the snoop can proceed. As this transition takes a significant amount of time, Arm recommends that MEM_RET is only used when other coherent agents are also idle.

DL1ONLY MEM_RET

In this PPU mode, the L2 cache RAMs are powered down and their content is not retained. Therefore this operating mode is equivalent to OFF mode. DL1ONLY MEM_RET is provided for consistency so that a system can choose to go to the MEM_RET state without needing to know the current operating mode.

DL1ONLY MEM_RET_EMU, FULL RAM MEM_RET_EMU, 1/2 RAM MEM_RET_EMU

In this PPU mode, the cluster logic including the L2 cache RAMs, L2 duplicate L1 tag RAMs, LCU duplicate L1 tag RAMs remains powered.

Individual cluster power modes such as ON, OFF, OFF_EMU, MEM_RET_EMU, DBG_RECOV, and WARM_RST are also considered to be PPU modes. See [5.9 Cluster power modes](#) on page 68 for more information on the description of these power modes.

5.11.1 Rules governing cluster PPU mode transitions

For the cluster *Power Policy Unit* (PPU) mode transitions, there is a set of rules that governs the transitions from each PPU mode. The PPUs are aware of these rules, so there is no requirement for the *System Control Processor* (SCP) or a core within the Cortex®-R82 processor through a Utility bus loopback connection to explicitly program these rules into the PPU.

The following rules govern all transitions between the PPU modes:

- When transitioning from OFF to ON, any supported operating mode can be targeted.
- Transitions between operating modes only happen in the ON power mode.
- Switching between DL1ONLY and FULL RAM ON can be direct or through ½ RAM ON.
- The operating mode is maintained when moving from ON to MEM_RET power mode.

5.11.2 PPU mode transition behavior

Where there is a transition between the *Power Policy Unit* (PPU) modes, the Cortex®-R82 processor cluster logic automatically performs a series of actions before accepting a new PPU mode.

The following table shows the allowed transitions between the cluster PPU modes and the associated actions.

**Note**

For each of the PPU mode transitions shown in the following table, additional actions (which are technology and implementation dependent) must be performed. These actions are carried out by partner implemented logic as part of the *Power Control State Machine (PCSM)*.

Table 5-5: Cluster PPU transition behavior

Start PPU mode	End PPU mode	Cortex®-R82 processor behavior
OFF	MEM_RET	No functional change.
OFF / OFF_EMU	ON	The L2 cache, L2 duplicate L1 tag RAMs, and <i>LLRAM Coherency Unit</i> (LCU) duplicate L1 tag RAMs are initialized, and the cluster is brought into coherency with the rest of the system. Cluster System register fields (except the debug, trace, and RAS register fields) which reset to architecturally UNKNOWN values are initialized to fixed values.
OFF_EMU	OFF	No functional change.
OFF_EMU	MEM_RET_EMU	No functional change.
MEM_RET / MEM_RET_EMU	ON	The L2 duplicate L1 tag RAMs and LCU duplicate L1 tag RAMs are initialized. Cluster System register fields (except the debug, trace, and RAS register fields) which reset to architecturally UNKNOWN values are initialized to fixed values.
MEM_RET_EMU	MEM_RET	No functional change.
ON	OFF / OFF_EMU	If there is any ongoing memory access, the request is denied. L2 cache allocation disabled, L2 cache cleaned and invalidated. The cluster is removed from system coherency.
ON	MEM_RET / MEM_RET_EMU	If there is any ongoing memory access, the request is denied.
ON	WARM_RST	Cluster System registers (excluding the debug, trace, and RAS registers) are reset.
WARM_RST	ON	Relevant ways in L2 cache are invalidated. Cluster System register fields (except the debug, trace, and RAS register fields) which reset to architecturally UNKNOWN values are initialized to fixed values.
Any mode	DEBUG_RECOV	Transition accepted immediately. Warm reset or Cold reset is applied depending on the value of DBG_RECOV_PORST_EN
DEBUG_RECOV	ON	Reset is deasserted. MACP and ACELS interfaces are enabled.
DL1ONLY	½ RAM	L2 cache tag RAM ways 0-3 invalidated. Cache lookup and allocation enabled for ways 0-3.
DL1ONLY	FULL RAM	L2 cache tag RAM ways 0-7 invalidated. Cache lookup and allocation enabled for ways 0-7.
½ RAM	DL1ONLY	L2 cache tag RAM ways 0-3 allocation is prevented. Ways 0-3 are cleaned of any dirty lines and then cache lookup is disabled.
½ RAM	FULL RAM	L2 cache tag RAM ways 4-7 invalidated. Cache lookup and allocation enabled for ways 4-7.
FULL RAM	½ RAM	L2 cache tag RAM ways 4-7 allocation is prevented. Ways 4-7 are cleaned of any dirty lines and then cache lookup is disabled.

Start PPU mode	End PPU mode	Cortex®-R82 processor behavior
FULL RAM	DL1ONLY	L2 cache tag RAM ways 0-7 allocation is prevented. Ways 0-7 are cleaned of any dirty lines and then cache lookup is disabled.

5.11.3 DebugBlock power modes

The DebugBlock supports only two power modes, ON and OFF. There is no *Power Policy Unit* (PPU) in the Cortex®-R82 processor for the DebugBlock. Instead, the DebugBlock has a Q-Channel interface for providing power control to the DebugBlock power domain.

When the DebugBlock is in the OFF mode, the DebugBlock does not initiate any accesses and all APB accesses to the DebugBlock receive a PSLVERR response.

5.12 Cluster powerdown

The cluster is taken out of coherency automatically when it is powered down. No software sequence is required.

After receiving the request to enter powerdown mode from the power controller, the Cortex®-R82 processor cleans and invalidates the L2 cache and communicates with the interconnect to disable snoops into the cluster. All cores must be in the OFF mode before the cluster is powered down.



Note

To support automated removal from coherency, the interconnect must support the SYSCOREQM and SYSOCOACKM handshake protocol. If not, the SoC is responsible for programming the interconnect to remove the cluster from coherency.

5.13 Cluster power mode and core power mode dependencies

There are some dependencies between the core and cluster power domains to ensure that correct operation is maintained.

The core and cluster power mode dependencies are the following:

- If a core ON request is made while the cluster is not in an ON mode, then the core request stalls until the cluster has reached the appropriate state.
- If a core is requested to go from WARM_RST to ON, the core request stalls until the cluster has transitioned from WARM_RST to ON.
- If a core is requested to go from DBG_RECOV to ON, the core request stalls until the cluster has transitioned from DBG_RECOV to ON.

- If the cluster is requested to go to MEM_RET or OFF while not all cores are OFF, then the cluster request is denied.
- If the cluster is requested to go from WARM_RST to ON, the cluster request is denied unless all the cores are in OFF, OFF_EMU, or WARM_RST.
- If the cluster is requested to go from DBG_RECOV to ON, the cluster request is denied unless all the cores are in OFF, OFF_EMU, or DBG_RECOV.
- If the cluster is requested to go to MEM_RET_EMU or OFF_EMU while not all cores are OFF or OFF_EMU, then the cluster request is denied.

6. Power and reset control with PPU

This chapter describes how to control the power modes and reset behavior of the Cortex®-R82 processor by using the *Power Policy Units* (PPUs).

6.1 The Power Policy Unit

The *Power Policy Units* (PPUs) enable control over the power modes of the Cortex®-R82 processor. The cluster PPU allows control over the cluster power and operating modes. Per-core PPU allows control over the individual core power modes.

A PPU is a standard component for abstracting the low-level hardware control signaling to software-controlled power domain policy. This allows the external agent to focus on the power modes it wants to achieve without being concerned about the intermediate power modes.

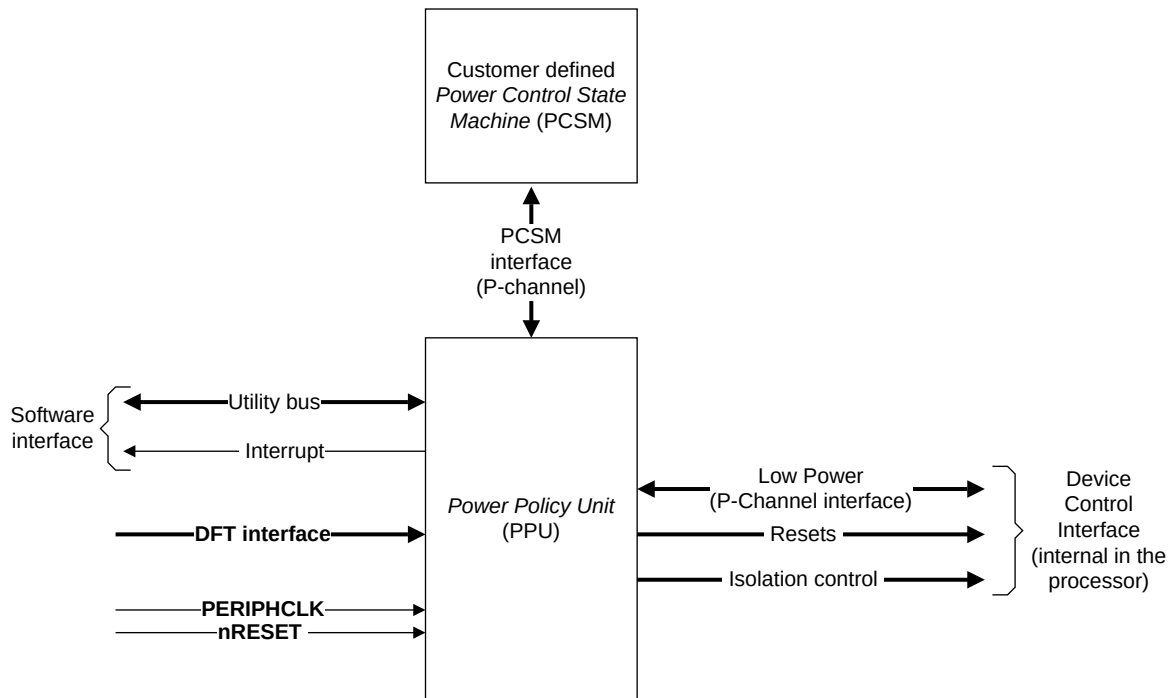
The implementation process automatically creates the PPU for the cluster and each core. Each PPU has a set of memory-mapped control registers which are accessed using the Utility bus.

The PPU can provide autonomous control of power modes with a range of modes. A component in the system such as a *System Control Processor* (SCP) can program the PPU over the Utility bus to set the appropriate power policy.

If your system does not have an SCP component, the Utility bus can be connected to one of the Cortex®-R82 processor ports or to your SoC interconnect, and one of the cores within the Cortex®-R82 processor can program the PPU. For information on the loopback address mapping through the interconnect, see [8.11 Utility bus](#) on page 176. For information on potential limitations of not using an SCP or not implementing a Utility bus loopback connection, see [6.9 Implications of not having a System Control Processor](#) on page 93.

The PPU controls the low-level details of powering up, powering down, or resetting domains as necessary to implement the requested policy. The hardware performs any actions necessary to reach the requested power mode, such as gating clocks, flushing caches, or disabling coherency.

The following figure shows the Cortex®-R82 processor PPU interfaces. All interfaces are external to the Cortex®-R82 processor apart from the Device Control interface.

Figure 6-1: PPU interfaces

The cluster PPU and per-core PPUs have the following main interfaces:

Software interface

The programming interface for the PPUs is accessed through the Utility bus. A set of PPU registers controls setting high-level policy control and configuration.

Device Control interface

The Device Control interface is the internal interface that connects to each of the cluster and core power domains. The Device Control interface provides low-level power control and ensures device quiescence.

The Device Control interface includes:

- The device interface that consists of one or more P-Channels.
- The control interface that includes resets and isolation control.



Note

Some of the Device Control interface signals are exported outside of the Cortex®-R82 processor to allow the control of other components that may be in the same power domain.

PCSM interface

The *Power Control State Machine* (PCSM) interface is an external interface for controlling low-level technology specific power switch and retention controls. There are separate PCSM interfaces for the cluster and each core instantiated in the Cortex®-R82 processor.

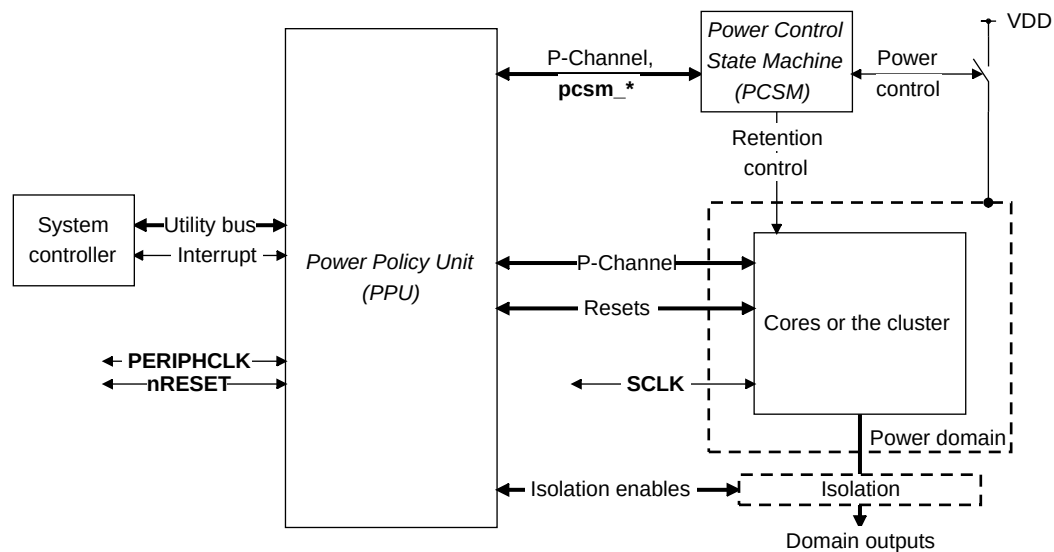


Note

The PCSM interface is always present even if your system does not implement some Cortex®-R82 power domains for cluster or core. For information on how to connect P-Channel signals in such cases, see P-Channel signals in *Arm® Cortex®-R82 Processor Configuration and Integration Manual*.

The following figure shows a high-level illustration of how the PPU and PCSM controls connect to each other and to a power-gated domain. The dotted lines indicate the implementation-dependent components and signal connections.

Figure 6-2: PPU connections to a power-gated domain



All the PPUs within the Cortex®-R82 processor are pre-built and are based on the CoreLink PCK600 Power Policy Units.



Note

The CoreLink PCK600 does not require a separate license. It is delivered as part of the Cortex®-R82 processor.

6.2 PPU operation

The *Power Policy Unit* (PPU) controls all of the Cortex®-R82 processor cluster and core power modes (ON, OFF, OFF_EMU, FUNC_RET, FULL_RET, MEM_RET, MEM_RET_EMU, WARM_RST, and DBG_RECOV). It has extensive support to reflect the various combinations of logic and memory power states into which a domain can be set.

Your software can program a PPU to set a PPU mode in one of two ways:

Static policy

A request to enter a PPU mode directly.

Dynamic policy

Sets a minimum mode, so the PPU can autonomously change the PPU mode above this mode depending on hardware inputs.

Each PPU contains a state machine representation of its supported PPU mode transitions. Therefore, a PPU can be programmed to target any supported PPU mode and the route taken follows the permissible route, passing through any intermediate PPU modes.

Each of the PPU has an interrupt output signal that indicates events such as the completion of power mode transitions and the completion of operating mode transitions. For the cluster, this is CLUSTERPPUIRQ and for the cores these are COREPPUIRQ[<m>], where <m> is the core instance number. These interrupts can be targeted to the agent which programs the PPU to avoid polling status registers.

A *System Control Processor* (SCP) or a core in the Cortex®-R82 processor programs the PPU through the Utility bus based on its current system requirements. The PPU modes are programmed using registers within the PPU. When a PPU mode is programmed into the PPU registers, this PPU mode is requested internally within the Cortex®-R82 cluster. The relevant cluster logic takes the necessary steps to enter this mode such as cleaning dirty cache lines from the caches when powering off. When the logic is ready to enter the required mode, the change in power state is requested on the relevant *Power Control State Machine* (PCSM) interface. When the power state transition is complete, the PCSM is responsible for changing the power state for the relevant logic and signaling to the PPU.

If the relevant logic is required to be isolated or removed from isolation this is signaled using the relevant ISOLATE signals (CLUSTERISOLATE_n, L2RAMISOLATE_n, FPSIMDISOLATE_n, and COREISOLATE_n). As the isolation signals do not have an acknowledge signal, the state of the isolation cells is changed at a software-dependent time later after a change of the ISOLATE signals. The latency time on the ISOLATE signal change can be configured by PPU registers. The PPU logic ensures that the isolation and power state change are requested in the required order for the requested PPU mode transition.

6.2.1 Implicit resets from PPU mode change

Transitions between power down modes and powered modes include an implicit internal reset of the powered off logic. This internal reset is managed by the *Power Policy Unit* (PPU) mode

controlling the transition between the two modes and does not require an external signal to be asserted or explicit programming of the PPU.

A transition from an Off mode to a power mode includes a Cold reset of the logic that was powered off, where both functional logic and debug logic is reset.

A transition from an Emulated off mode to a power mode includes a Warm reset of the logic that was emulated as powered off, where the functional logic is reset but the Debug, Trace and *Reliability, Availability, and Serviceability* (RAS) state is not reset.

6.3 Utility bus accesses

All of the *Power Policy Unit* (PPU) control and data registers are accessed using the memory mapped Utility bus. The Utility bus is implemented as a 64-bit AMBA® 5 AXI subordinate port.

Accesses to PPU registers over the Utility bus must match the size of the register they are accessing, either 32 bits or 64 bits. Any access with sizes other than 32-bit or 64-bit gets an SLVERR response from the Utility bus.

Cores within the Cortex®-R82 processor cannot access to the PPU registers directly. Instead, you must either use a *System Control Processor* (SCP), or configure your interconnect to provide a loopback address mapping for the cores to access the Utility bus. See [6.9 Implications of not having a System Control Processor](#) on page 93 for information on how to program the PPU from within the Cortex®-R82 processor.

The registers for the cluster PPU and each of the core PPU are grouped on separate 64KB page boundaries allowing access control to be enforced by the memory management. The DESNSE_CS_ADDR_MAP options allows for packing for 4KB page boundaries.

Only secure accesses are allowed on the Utility bus to access PPU registers. Any Non-secure access gets an SLVERR response from the Utility bus.

6.4 Encodings for cluster power modes and operating modes

The cluster *Power Policy Unit* (PPU) power policy register CLUSTERPPU_PWPR uses power mode and operating mode encodings to set various power conditions for the cluster. For example, the values at the register bitfields CLUSTERPPU_PWPR.PWR_POLICY and CLUSTERPPU_PWPR.OP_POLICY set the power mode and operating mode of the cluster.

The following table shows the Cortex®-R82 processor cluster level power mode encodings.



Note

- Priority is an architectural concept as defined in [Arm® Power Policy Unit Architecture Specification](#).
- INPUTDOMAINSTATE[3:0] and OUTPUTDOMAINSTATE[3:0] encodings are the same as the CLUSTERPPU_PWPR.PWR_POLICY[3:0] encodings.

Table 6-1: Power mode enumeration for the cluster

Power mode	CLUSTERPPU_PWPR.PWR_POLICY[3:0]	CLUSTERPCSMPSTATE[3:0]	CLUSTERPPUHWSTAT[15:0]	Priority
OFF	0x0	0x0	0x0001	Low
OFF_EMU	0x1	0x8	0x0002	-
MEM_RET	0x2	0x2	0x0004	-
MEM_RET_EMU	0x3	0x8	0x0008	-
ON	0x8	0x8	0x0100	-
WARM_RST	0x9	0x8	0x0200	-
DBG_RECOV	0xA	0x8	0x0400	High

The following table shows the Cortex®-R82 processor cluster operating mode encodings for CLUSTERPCSMPSTATE[5:4] and CLUSTERPPUHWSTAT[23:16].

Table 6-2: Operating mode enumeration for the cluster

Operating mode	Short name	CLUSTERPPU_PWPR.OP_POLICY	CLUSTERPCSMPSTATE[5:4]	CLUSTERPPUHWSTAT[23:16]	Priority
L2 duplicate L1 tag RAM and LLRAM Coherency Unit (LCU) duplicate L1 tag RAM only	DL1ONLY	0x0	0x0	0x01	Low
Half L2 cache	½ RAM	0x1	0x1	0x02	Medium
Full L2 cache	FULL RAM	0x2	0x2	0x04	High

For information on these registers, see [Arm® Power Policy Unit Architecture Specification](#).

6.5 Encodings for core power modes

The core *Power Policy Unit* (PPU) power policy registers PPU_PWPR use power mode encodings to set various power modes for the cores.

The following table shows the power mode encodings for the cores in the Cortex®-R82 processor.



In the following table <m> is the core instance number and Priority is an architectural concept as defined in [Arm® Power Policy Unit Architecture Specification](#).

Table 6-3: Power mode enumeration for the cores

Power mode	PPU_PWPR.PWR_POLICY	CORE<m>PCSMSTATE[3:0]	CORE<m>PPUHWSTAT[15:0]	Priority
OFF	0x0	0x0	0x0001	Low
OFF_EMU	0x1	0x8	0x0002	-
FULL_RET	0x5	0x5	0x0020	-
FUNC_RET	0x6	0x7	0x0080	-
ON	0x8	0x8	0x0100	-
WARM_RST	0x9	0x8	0x0200	-
DBG_RECOV	0xA	0x8	0x0400	High

6.6 Programming sequences for the cluster and the core

Each request for a change in *Power Policy Unit* (PPU) mode triggers a power state request on the *Power Control State Machine* (PCSM) interface for the respective core or cluster. The PCSM must accept the change request in power state before the PPU mode can change.

A request on the PCSM interface can be accepted automatically for designs where:

- There is no active power management at all.
- The cluster is being analyzed in a non-power aware environment.

6.6.1 Programming sequence to bring the cluster and cores from Off to On mode

Use the following steps to program the *Power Policy Unit* (PPU) for the cluster and each core to request a change in PPU mode from Off to On.

About this task

This task uses the PPU static policy to request a single mode transition. <m> is the core instance number.

Procedure

1. Write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA. Write to the cluster register CLUSTERPPU_PWPR, address 0x010000, value 0x00020008.

This sets the power mode policy to ON and the operating mode policy to FULL RAM.

2. Poll the cluster CLUSTERPPU_PWSR register, address 0x010008, until the value read matches the value written to the PPU_PWPR register.
Alternatively, you can enable the PPU interrupt and wait for the interrupt from the PPU to read the cluster CLUSTERPPU_PWSR register.



Any update to cluster or core PPU registers, including the PPU Interrupt Mask Register, requires a write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA.

The *Power Control State Machine* (PCSM) for the cluster must respond back to the PPU mode request before the mode is accepted.

3. Write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA. Write to the core PPU_PWPR register, for core <m>, address 0x<m>40000, value 0x00000008.
4. Poll the core PPU_PWSR register for core <m>, address 0x<m>40008, until the value read matches the value written to the PPU_PWPR register.
Alternatively, you can enable the PPU interrupt and wait for the interrupt from the PPU to read the core PPU_PWSR register.

Each core PCSM must respond back to the PPU mode request before the mode is accepted.

6.6.2 Programming sequence to bring the cluster and cores from On to Off mode

Use the following steps to program the *Power Policy Unit* (PPU) for the cluster and each core to request a change in PPU mode from On to Off.

About this task

This task uses the PPU static policy to request a single mode transition. <m> is the core instance number.

Procedure

1. Software running on the core sets the IMP_CPUPWRCTLR_EL1.PWRDN bit to 1, then executes a `WFI` instruction. For the full sequence see, [5.8 Core powerdown](#) on page 67.
2. Write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA. Write to the core PPU_PWPR for core <m>, address 0x<m>40000, value 0x00000000.
3. Poll the core PPU_PWSR register for core <m>, address 0x<m>40008, until the value read matches the value written to the PPU_PWPR register.
Alternatively, you can enable the PPU interrupt and wait for the interrupt from the PPU to read the core PPU_PWSR register.

The core *Power Control State Machine* (PCSM) must respond back to the PPU mode request before the mode is accepted.

4. Write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA. Write to the cluster CLUSTERPPU_PWPR register, address 0x010000, value 0x00000000.
5. Poll the cluster CLUSTERPPU_PWSR register, address 0x010008, until the value read matches the value written to the CLUSTERPPU_PWPR register.
Alternatively, you can enable the PPU interrupt and wait for the interrupt from the PPU to read the cluster CLUSTERPPU_PWSR register.

The cluster PCSM must respond back to the PPU mode request before the mode is accepted.

6.6.3 Programming sequence for an interrupt controller to bring cores and cluster to On or Off mode

The following sequence makes it possible for the cluster and cores in the Cortex®-R82 processor to automatically power down when software running on the processor has nothing more to do. It also enables the cluster and cores to automatically power up when they receive the asserted signal COREWAKEREQUEST[<m>] from an interrupt controller.

About this task

This task uses the PPU dynamic policy to request automatic transitions. <m> is the core instance number.

Procedure

1. Write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA. Write to the cluster CLUSTERPPU_PWPR register, address 0x010000, value 0x01000100.
2. Write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA. Write to the core PPU_PWPR for core <m>, address 0x<m>40000, value 0x00000100.
3. To power up core <m> or power down core <m>, see the following steps.

Choice

To power up core <m>

To power down core <m>

Step

Assert the COREWAKEREQUEST[<m>] signal.

Software on the core sets the IMP_CPUPWRCTLR_EL1.PWRDN bit to 1, then executes a WFI instruction. The full power down sequence in [5.8 Core powerdown](#) on page 67 should be respected.

When all cores are powered off, the cluster will power off with no additional action needed.



Note

The signal COREWAKEREQUEST[<m>] is level sensitive.

6.7 Explicit resetting of cluster and cores and debug recovery

You can reset part or all of the Cortex®-R82 processor in various ways. This section describes the sequences to reset part or all of the Cortex®-R82 processor.



Note

- You must follow the sequences exactly.
- The *Power Policy Units* (PPUs) and associated logic prevents unsupported transactions from occurring.
- The examples that refer to addresses of PPU registers, assume that the parameter `DENSE_CS_ADDR_MAP = 0`
- The `WARM_RST` and `DBG_RECOV` power modes do not have an associated operating mode. Therefore before entering these power modes, the current cluster operating mode must be saved. This ensures that the same operating mode can be restored when leaving the power states.

Powerup (Cold) reset

This reset must be done the first time that the cluster is powered up. It resets all the Cortex®-R82 processor including the PPU.



Note

This procedure can be achieved by either programming the PPU over the Utility bus or configuring the `PPU_RST_STATE` parameter to 1.

1. Assert the `nRESET` signal for a minimum of 10 or more `PERIPHCLK` cycles.
2. Deassert the `nRESET` signal.
3. Program the PPU for the cluster to On mode, see [6.6.1 Programming sequence to bring the cluster and cores from Off to On mode](#) on page 86.
4. Program the PPU for each core to On mode, see [6.6.1 Programming sequence to bring the cluster and cores from Off to On mode](#) on page 86.



Note

Steps 3 and 4 are not needed if the Cortex®-R82 processor is configured with `PPU_RST_STATE = 1`. See [6.9 Implications of not having a System Control Processor](#) on page 93 for more information.

Software initiated Warm reset of an individual core



Note

- This procedure does not require any access on the Utility bus.
- Interrupts and other external accesses are expected to be redirected or paused from the point this procedure is initiated until the core restarts execution.

1. Use software running on the core to program the RMR_EL2.RR register bit.
2. Execute a `WFI` instruction.

Software initiated Cold or Warm reset of the cluster (excluding the PPUs)

For the Cold reset case, power is also removed from the cluster during this sequence.



Note

This procedure requires an external SCP connected via the Utility bus.

1. Use software running on each core to set the IMP_CPUPWRCTLR_EL1.PWRDN bit.
2. Use software running on each core to execute a `WFI` instruction.
3. Program the core's PPU, and then the cluster PPU, to Off mode (Cold reset) or Emulated off mode (Warm reset) as per [6.6.2 Programming sequence to bring the cluster and cores from On to Off mode](#) on page 87.



Note

- The cluster Off mode can only be entered if all the cores are in Off mode.
- To use Emulated off in the cluster, core, or both a value of `0x00000001` should be written to CLUSTERPPU_PWPR and PPU_PWPR respectively.

4. To transition from Emulated off mode back to On, first program the cluster PPU to ON. Then program the Core PPUs to ON.
5. To transition from Off mode to On mode, see [6.6.1 Programming sequence to bring the cluster and cores from Off to On mode](#) on page 86.

Using WARM_RST mode to reset the cluster (excluding the PPUs)

This procedure can be used to recover from a watchdog timeout or similar situations.



Note

This procedure can be achieved by the *System Control Processor* (SCP) over the Utility bus. If your system does not have an SCP, you have to enable an external agent such as debugger that has memory mapped access to the Utility bus.

1. Ensure that the cluster is in On mode and the cores are either in On mode, Off mode, or Emulated off mode. Read the PPU_PWSR for the cluster to determine the current cluster operating mode.

**Note**

Before every register write, the user must first write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA.

2. For any of the cores that are in On mode, write to the core PPU_PWPR for core <n>, address 0x<n>40000, value 0x00000009. This sets the core to the WARM_RST power mode.
3. Write to the cluster PPU_PWPR, address 0x010000, value 0x00000009. This sets the cluster to the WARM_RST power mode.
4. Write to the cluster PPU_PWPR, address 0x010000, value 0x000<p>0008, where <p> is the operating mode value read in step 1. This sets the cluster to the ON power mode.
5. For each core that is in WARM_RST, write to the core PPU_PWPR register, for core <n>, address 0x<n>40000, value 0x00000008. This puts each core back to the ON power mode.

**Note**

After each of the *PPU_PWPR is configured, the corresponding *PPU_PWSR should be polled until the value read matches the value written to the *PPU_PWPR register. Alternatively, you can enable the PPU interrupt and wait for the interrupt from the PPU to read the cluster *PPU_PWSR register. Addresses for these can be found in [B.1.2.2.3 PPU_PWSR, Power Status Register](#) on page 1481. The corresponding PCSM must respond back to the PPU mode request before the mode is accepted.

Reset of the cluster (excluding the PPU), retaining cache contents for debug

This can be either a Warm reset or Cold reset, depending on the setting of the PPU_PTCR.DBG_RECOV_PORST_EN bit.

**Note**

This procedure can be achieved by the *System Control Processor* (SCP) over the Utility bus. If your system does not have an SCP, you have to enable an external agent such as debugger that has memory mapped access to the Utility bus.

**Note**

The value in PPU_PTCR.DBG_RECOV_PORST_EN bit must be the same for all PPU (the cluster and all the cores). Otherwise the results are **UNPREDICTABLE**.

1. Read the PPU_PWSR for the cluster and each core to determine which cores are powered up and what is the current cluster operating mode.

2. For any cores that are already OFF, you must ensure they are in a static OFF, or in a LOCKED OFF (PPU_PWPR.LOCK_EN = 1 for core <m>, address 0x<m>40000, bit 12) state to ensure they do not power up during this process.
3. Check the cluster operating mode and ensure it is in a static configuration so that the operating mode does not change after this step.

**Note**

Before every register write, the user must first write to the cluster safety write key register CLUSTERSAFETY_WRITEKEY, address 0x080060, value 0x000000BA.

4. Write to the core PPU_PWPR for core <m>, address 0x<n>40000, value 0x0000000A. This sets the core to the DBG_RECOV power mode.
 - If PPU_PTCR.DBG_RECOV_PORST_EN = 1, any core not in the OFF mode must be put in the DBG_RECOV mode.
 - If PPU_PTCR.DBG_RECOV_PORST_EN = 0, any core not in the OFF or OFF_EMU mode must be put in the DBG_RECOV mode.
5. Write to the cluster PPU_PWPR, address 0x010000, value 0x0000000A. This sets the cluster to the DBG_RECOV power mode.
6. Write to the cluster PPU_PWPR, address 0x010000, value 0x000<p>0008, where <p> is the operating mode value read in step 1. This sets the cluster to the ON power mode.
7. For each core that is in DBG_RECOV, write to the core PPU_PWPR register, for core <m>, address 0x<n>40000, value 0x00000008. This sets each core back to the ON power mode.

- After each of the *PPU_PWPR is configured, the corresponding *PPU_PWSR should be polled until the value read matches the value written to the *PPU_PWPR register. Alternatively, you can enable the PPU interrupt and wait for the interrupt from the PPU to read the cluster *PPU_PWSR register. Addresses for these can be found in [B.1.2.2.3 PPU_PWSR, Power Status Register](#) on page 1481. The corresponding PCSM must respond back to the PPU mode request before the mode is accepted.

**Note**

- If your system does not have a *System Control Processor* (SCP), some of the sequences above might not be available. For information on potential limitations of not using an SCP or not implementing a Utility bus loopback connection, see [6.9 Implications of not having a System Control Processor](#) on page 93.
- To control transitions through in and out of DBG_RECOV and WARM_RST power modes, it is expected that a debugger will have memory mapped access to the Utility bus. Therefore, even if your system does not have an SCP, it should provide memory mapped access to the Utility bus for debug control.

6.8 ECC errors during power transitions

It is possible to get *Error Correcting Code* (ECC) errors in the RAMs during a power transition. These ECC errors could happen during the software sequence shortly before the hardware sequence starts. Alternatively, these ECC errors could happen during the hardware sequence when the L1 or L2 cache is cleaned and invalidated as part of the On to Off power mode transition for a core or the cluster.

If the *Reliability, Availability, and Serviceability* (RAS) interrupts are enabled and if RAS errors occur which cause RAS interrupts while the cluster is powering down, then the RAS interrupts prevent the core or the cluster from powering down to prevent loss of information.

The RAS interrupts can be disabled. If the RAS interrupts are disabled, then even though a RAS error occurs during a power down transition, the core or cluster can still continue with the powering down.

If the RAS interrupts are enabled, then you must ensure that your system has either of the following:

- An external system level error manager which has the ability to read and write RAS records through the Utility bus.
- Nominated one (or more) Cortex®-R82 processor core to be responsible for handling the RAS interrupts. In this case, the nominated core and the cluster must be always powered on and available to handle the RAS interrupts. The nominated core must also have access to the RAS records through a Utility bus loopback.



In systems where the RAS interrupts are enabled, if there is no external system level error manager and if the nominated core for handling the RAS interrupts is powered down, this can lead to an incomplete state. This is because a denied power transaction will leave the nominated core powered on but stuck in WFI. In this case, the RAS interrupts will remain active but not be serviced.

Although the ECC errors are reported in the RAS error record registers, once the cluster or core is powered down, the RAS registers are no longer accessible. If the RAS registers are reporting an error and the RAS interrupts are enabled, the following sequence happens:

1. The RAS interrupt signals (nCOREERRIRQ, nCOMPLEXERRIRQ, nCOREFAULTIRQ, nCOMPLEXFAULTIRQ or nCOMPLEXCRITIRQ) for the appropriate core or cluster are asserted.
2. If the *Power Policy Unit* (PPU) is requesting a transition to an OFF power mode, then the request to OFF power mode is denied.
3. The error interrupts must be sent to either an external system level error manager or a nominated Cortex®-R82 processor core which is always on. The external system level error manager or a nominated core then reads and writes the relevant RAS error record registers through the Utility bus in order to clear the record of the reported fault or error. Once the error record has been cleared, the PPU will be able to request the OFF power mode again.

6.9 Implications of not having a System Control Processor

The global `PPU_RST_STATE` parameter defines the Cold reset state for the core and cluster *Power Policy Units* (PPUs).

`PPU_RST_STATE` supports the following values:

- | | |
|----------|---|
| 0 | Cluster PPU and all core PPUs reset to Off. |
| 1 | Cluster PPU and all core PPUs reset to On. |

Setting the `PPU_RST_STATE` to 0 ensures the whole Cortex®-R82 processor is Off until a *System Control Processor* (SCP) powers it On. You must use this option only if your system includes an SCP that is connected to the Cortex®-R82 processor Utility bus.

Setting the `PPU_RST_STATE` to 1 allows all the PPUs to transition from Off to On automatically after reset deassertion, without any programming by the SCP. If required, any core within the Cortex®-R82 processor can still access the PPUs through the *Main Manager* (MM) port or the *Shared Peripheral Port* (SPP) that is connected via a loopback to the Utility bus. For an example of a loopback address mapping, see [8.11 Utility bus](#) on page 176.

You must consider the following when configuring the `PPU_RST_STATE`:

- If your system has an SCP, `PPU_RST_STATE` can be configured to either 0 or 1.
- If your system does not have an SCP, configure the `PPU_RST_STATE` to 1. Configuring the `PPU_RST_STATE` to 0 will leave your Cortex®-R82 cluster permanently to an Off state, if no agent is able to program the PPUs.

If your system does not have an SCP or a loopback connection to the Utility bus, there is no way to program the PPUs. This means several things such as dynamic OPMODE transitions will not work. Also, some of the sequences in [6.7 Explicit resetting of cluster and cores and debug recovery](#) on page 88 cannot happen. For some of those cases you have to enable an external agent such as debugger that is connected to the Utility bus in your system to enable the cluster and cores to On mode to avoid a system deadlock.

6.10 PPU and reset management register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary tables for the core and cluster *Power Policy Unit* (PPU) registers in [B.1.1.2 External PPU registers summary](#) on page 1417 and [B.1.1.3 External CLUSTERPPU registers summary](#) on page 1419.

7. Initialization

This chapter describes considerations for initializing the Cortex®-R82 processor.

7.1 Initializing the Cortex®-R82 processor

You need to consider several things before you can run any program on the Cortex®-R82 processor. This includes, initializing all programmer-visible registers, enabling the *Memory Protection Unit* (MPU), enabling the *Floating Point Unit* (FPU), invalidating the caches, and programming the *Power Policy Units* (PPUs).

Initializing the registers

Most of the architectural registers in the Cortex®-R82 processor, such as X0-X30 and S0-S31 and D0-D31 when Advanced SIMD is included, do not have an architecturally defined reset value, that is they have an **UNKNOWN** value after reset. The Cortex®-R82 processor includes hardware that automatically initializes all programmer-visible registers to a fixed value out of reset, including those which do not have an architecturally defined reset value.



The *Process State* (PSTATE) and some System register fields are given a known value on reset, see the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

Each core within the Cortex®-R82 processor includes hardware that initializes all core programmable-visible registers after a core reset, unless the reset is for debug recovery.

Similarly, the Cortex®-R82 processor includes hardware that initializes all cluster programmer-visible registers after a cluster reset, unless the reset is for debug recovery.



You must not assume that these initialization values can be depended on. The initialization values must still be considered as **UNKNOWN**, and that they could change in subsequent processor releases. This means that software must always initialize the programmer-visible registers that are defined as having **UNKNOWN** reset values before using them.

In addition, before you run an application, remember to:

- Program particular values into various registers, for example, Stack Pointers.
- Enable various features, for example, error reporting.
- Enable Region Registers controlling *Data Tightly Coupled Memory* (DTCM), *Tightly Coupled Memory* (ITCM), *Low-latency Peripheral Port* (LLPP), and *Low-latency RAM* (LLRAM) regions.
- Program particular values into memory, for example, the TCMs.

Enabling the MPU

To make use of the programmable regions in the *Memory Protection Unit* (MPU), you must configure and enable any regions you wish to use. The Cortex®-R82 processor supports direct programming through System register operations.

Before you can use the programmable regions, you must:

- Program the required regions for your memory map
- Enable the programmed regions
- Enable translation in the *System Control Register* (SCTLR), SCTLR_EL1.M and SCTLR_EL2.M

If using programmable regions the MPU should not be enabled until the regions are programmed and active. It is also possible to make use of default attributes without enabling any programmable regions. Please refer to the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more details.

Enabling the MMU

To use *Memory Management Unit* (MMU) at EL1/0 you must:

- Configure the pagetables in the memory
- Configure the TTBR register to point to the pagetables
- Configure the VTCR_EL2.MSA bit to select VMSA in EL1/0
- Configure the HCR_EL2 if virtualization needs to be enabled
- Enable the *System Control Register* (SCTLR) SCTLR_EL1.M

Enabling the FPU

You must enable the *Floating Point Unit* (FPU) before floating-point or Advanced SIMD instructions can be executed.

Enable the FPU as follows:

- Enable access to the FPU in the *Architectural Feature Access Control Register* (CPACR_EL1) by setting the FPEN field.
- Disable trapping of FPU instructions in the *Architectural Feature Trap Register* (CPTR_EL2) by resetting the TFP bit.

Invalidating the caches

The Cortex®-R82 processor L1 instruction and L1 data caches and, if implemented, L2 cache are automatically invalidated after a reset.

This operation can never report any ECC errors.



Each core within the Cortex®-R82 processor includes hardware that invalidates all core caches after a core reset, only if the reset is not because of an MBIST request or for debug recovery and the core power mode transitions from any OFF mode to any ON mode.

Similarly, the Cortex®-R82 processor includes hardware that invalidates the L2 cache, L2 duplicate L1 tag RAMs, and *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs after a cluster reset, only if the reset is not because of an MBIST request or for debug recovery and the cluster power mode transitions from:

- Any OFF mode to any ON mode
- DL1ONLY ON to ½ RAM ON
- ½ RAM ON to FULL RAM ON

Programming the PPU

The PPUs control the power modes and reset behavior of the Cortex®-R82 processor.

You can control the cluster and core PPUs to reset to off or on with the global `PPU_RST_STATE` parameter. Depending on the value of the `PPU_RST_STATE` parameter, an external *System Control Processor* (SCP) or core 0 within the Cortex®-R82 processor programs the PPUs through the Utility bus and brings the Cortex®-R82 processor out of reset.

For more information on `PPU_RST_STATE` parameter, see [6.9 Implications of not having a System Control Processor](#) on page 93. For more information on the PPU programming sequences, see [6.6 Programming sequences for the cluster and the core](#) on page 86.

7.2 Initializing TCMs

Each core within the Cortex®-R82 processor has two optional *Tightly Coupled Memories* (TCMs): an *Instruction Tightly Coupled Memory* (ITCM) and a *Data Tightly Coupled Memory* (DTCM).

You can configure the Cortex®-R82 processor to:

- Enable the ITCM out of reset.
- Boot out of an initialized ITCM.

A core's TCMs can be initialized by that core directly or by an external agent. If an external agent is used, the Cortex®-R82 processor supports enabling initialization of a core's TCMs before that core completes its reset exception sequence and setting the reset vector address to point into the initialized ITCM.



Note

Before initializing the TCMs, ensure that both PDCLUSTER and PDCPU<m> power domains are ON, either by configuring the processor with `PPU_RST_STATE = 1` or by programming the *Power Policy Units* (PPUs).

7.2.1 Preloading TCMs

You can write data to the *Tightly Coupled Memories* (TCMs) by using either store instructions or the *ACE-Lite Subordinate* (ACELS) interface.

Depending on the method you choose, you might require:

- Particular hardware on the SoC that you are using, such as a DMA engine.
- Boot code.
- A debugger connected to the processor.

The methods to preload the TCMs include:

Memory copy with running boot code

The boot code includes a memory copy routine that reads data from an external memory, and writes it into the appropriate TCM. You must enable the TCM to do this.

Copy data from the Debug Communications Channel

The boot code includes a routine to read data from the *Debug Communications Channel* (DCC) and write it into the TCM. The debug host feeds the data for this operation into the DCC by writing to the appropriate registers on the processor APB debug interface.

Execute code in debug halt state

The debug host puts the Cortex®-R82 processor into debug halt state and then feeds instructions into it through the *External Debug Instruction Transfer Register* (EDITR). The Cortex®-R82 processor executes these instructions, as an alternative to using boot code in either of the two methods previously described.

DMA into TCM

The SoC includes a DMA device that reads data from a ROM and writes it to the TCMs through the ACELS interface.

7.2.2 Preloading TCMs with ECC

Before a RAM location is read with *Error Correcting Code* (ECC) protection enabled, the error code bits must be initialized.

If you included the internal RAM protection functionality in the Cortex®-R82 processor, the error code bits in the TCM RAM are present and not initialized by the Cortex®-R82 processor.

The Cortex®-R82 processor has separate controls of whether ECC checking is enabled and whether detected ECC errors are recorded and reported to the appropriate *Reliability, Availability, and Serviceability* (RAS) registers. For more information on the relevant registers see the MEMPROTEN fields in [A.2.2.46 IMP_MEMPROTCTLR_EL1, Memory Protection Control Register](#) on page 529 and [A.2.2.59 IMP_CLUSTERMEMPROTCTLR_EL1, Cluster Memory Protection Control Register](#) on page 573 and ED field in [B.1.2.1.2 ERR<n>CTLR, Error Record <n> Control Register, n = 0 - 9](#) on page 1424.

To update a TCM location without detecting an error, either the ECC checking or RAS reporting must be disabled or the write must be of the same width and aligned to the data chunk that the error scheme protects as described in this section.

**Note**

The reset values of IMP_MEMPROTCTLR_EL1.MEMPROTEN and ERR<n>CTLR.ED fields imply that errors are detected and, if possible, corrected, but are not recorded to RAS registers.

You can use the CFGITCMEN<m> signal to enable the *Instruction Tightly Coupled Memory* (ITCM) when leaving reset.

To initialize the TCM RAM without causing any spurious errors to be reported, follow these rules:

- If the *ACE-Lite Subordinate* (ACELS) interface is used to initialize a TCM with ECC enabled, AXI-Lite subordinate transactions must start at 128-bit aligned addresses, writing continuous blocks of memory of at least 64 bytes each and all bytes in the block enabled.
- If initialization is done by running code on the Cortex®-R82 processor, this is best done by a loop of stores that write to the whole of the TCM memory. For both TCMs use *Store Pair* (STP) instruction to 128-bit aligned addresses.
- If you are initializing the ITCM with your program code, ensure that:
 - At least 128 bytes past the end of your program are also initialized to a known value.
 - Any remaining uninitialized parts of the ITCM should be marked as Execute-never, by programming the appropriate *Memory Protection Unit* (MPU) regions or appropriate *Memory Management Unit* (MMU) pages.

7.2.3 Using TCMs from reset

You can use the CFGITCMENm signal to enable the *Instruction Tightly Coupled Memory* (ITCM) from reset, the CFGITCMBASEADDRm signal to select the ITCM base address, and the CFGRVBARADDRm to select the reset vector address to fall within the ITCM address range. This enables you to configure the processor to boot from *Tightly Coupled Memories* (TCMs) but, to do this, the TCMs must first be preloaded with the boot code.

The Cortex®-R82 processor has CPUHALTm input for each core that, when asserted, prevents the core from starting to execute instructions out of reset. This enables the TCMs to be preloaded before the core boots. If an external debug request is made before this input is deasserted, then the core waits until CPUHALTm has been deasserted, and then enters debug halt state before executing any instructions.

**Note**

ECC error reporting in the TCMs is suppressed when the processor core is in HALT (CPUHALTm = 1), except where an explicit read request targeting the TCMs has been received on the *ACE-Lite Subordinate* (ACELS) port.

The CPUHALTm input can be asserted while the processor is in reset to prevent the processor from fetching and executing instructions after coming out of reset. While the processor is halted in this way, the TCMs can be preloaded with the appropriate data. When the CPUHALTm input is deasserted, the processor starts fetching instructions from the reset vector address in the normal way.

**Note**

When CPUHALTm has been deasserted to start the processor fetching, it must not be asserted again except when the core is under core warm or powerup reset.

7.3 Initializing LLRAM

The Cortex®-R82 processor has an optional *Low-latency RAM* (LLRAM) manager interface that is shared among the cores within the cluster.

You can configure the Cortex®-R82 processor to:

- Enable the LLRAM out of reset.
- Boot out of an initialized LLRAM.

The LLRAM can be initialized by any core in the cluster directly or by an external agent. If an external agent is used, the Cortex®-R82 processor supports enabling initialization of LLRAM before a core in the cluster completes its reset exception sequence and setting the reset vector address to point into the initialized LLRAM.

**Note**

Before initializing the LLRAM:

- If using a DMA, ensure that the PDCLUSTER power domain is ON or PPU_RST_STATE is set to 1.
- If using memory copy or debugger, ensure that both PDCLUSTER and PDCPU<m> power domains are ON or PPU_RST_STATE is set to 1.

7.3.1 Preloading LLRAM

You can write data to the memory that is connected to the *Low-latency RAM* (LLRAM) port by using either store instructions or the *ACE-Lite Subordinate* (ACELS) interface.

Depending on the method you choose, you might require:

- Particular hardware on the SoC that you are using, such as a DMA engine.
- Boot code.
- A debugger connected to the processor.

The methods to preload the memory that is connected to the LLRAM port include:

Memory copy with running boot code

The boot code includes a memory copy routine that reads data from an external memory, and writes it into the memory that is connected to the LLRAM port. You must enable the LLRAM to do this.

Copy data from the Debug Communications Channel

The boot code includes a routine to read data from the *Debug Communications Channel* (DCC) and write it into the memory that is connected to the LLRAM port. The debug host feeds the data for this operation into the DCC by writing to the appropriate registers on the processor APB debug interface.

Execute code in debug halt state

The debug host puts the Cortex®-R82 processor into debug halt state and then feeds instructions into it through the *External Debug Instruction Transfer Register* (EDITR). The Cortex®-R82 processor executes these instructions, as an alternative to using boot code in either of the two methods previously described.

DMA into LLRAM

The SoC includes a DMA device that reads data from a ROM and writes it to the memory that is connected to the LLRAM port through the ACELS interface.

7.3.2 Using LLRAM from reset

You can use the CFGLLRAMEN signal to enable the *Low-latency LLRAM* (LLRAM) from reset, the CFGLLRAMBASEADDR signal to select the LLRAM base address, and the CFGRVBARADDRm to select the reset vector address to fall within the LLRAM address range. This enables you to configure the processor to boot from LLRAM but, to do this, the memory that is connected to the LLRAM port must first be preloaded with the boot code.

The Cortex®-R82 processor has CPUHALTm input for each core that, when asserted, prevents the core from starting to execute instructions out of reset. This enables the LLRAM to be preloaded before the core boots. If an external debug request is made before this input is deasserted, then the core waits until CPUHALTm has been deasserted, and then enters debug halt state before executing any instructions.

If the PDCPU<m> power domain is ON, the CPUHALTm input can be asserted while the processor is in reset to prevent the processor from fetching and executing instructions after coming out of reset. While the processor is halted in this way, the memory that is connected to the LLRAM can be preloaded with the appropriate data.

When the CPUHALTm input is deasserted, the processor starts fetching instructions from the reset vector address in the normal way.

**Note**

When CPUHALTm has been deasserted to start the processor fetching, it must not be asserted again except when the core is under core warm or powerup reset.

7.4 Disabling EL2

The Cortex®-R82 processor always boots into EL2.

If you do not need to use EL2 after the bootup process is complete, you can program the Cortex®-R82 processor and switch to EL1 so that it can never return to EL2 until the next reset. This involves setting all exceptions to be taken at EL1 and disabling `hvc` and the EL2-controlled *Memory Protection Unit* (MPU).

To disable EL2 and enter EL1:

- Program the ACTLR_EL2 register because it defaults to only allowing EL2 accesses. ACTLR_EL2 controls whether EL1 can access various processor resources. Other registers default to allowing accesses at EL1 from reset.
- Program the CPTR_EL2 register because it resets to **UNKNOWN** values. CPTR_EL2 controls whether various EL1 architectural features are trapped.
- Program the following EL2 generic timer registers to appropriate values because their fields reset to **UNKNOWN** values.
 - CNTHCTL_EL2 fields can control whether various EL0 and EL1 timer register accesses are trapped to EL2.
 - CNTHPS_CTL_EL2 fields control timer interrupts for the EL2 physical timer.
- Program the VTCR_EL2 register because it resets to **UNKNOWN** values. VTCR_EL2 controls the memory system translations.
- Program the rest of the HCR_EL2 register fields because they reset to **UNKNOWN** values. For example, disable all traps to EL2, disable stage 2 address translation, disable default cacheability, route interrupts to EL1, disable virtual interrupts.
- Set VBAR_EL1 to the correct location for the vector table.
- Disable the `hvc` instruction by setting HCR_EL2.HCD to 1.
- Set ELR_EL2 and SPSR_EL2 to point to the entry point and the desired state of the EL1 code and call ERET.

8. Memory system

This chapter describes the various memories and memory interfaces of the Cortex®-R82 processor.

8.1 About the memory system

The Cortex®-R82 processor memory system provides various memories and interfaces each tailored to different requirements.

The memory system includes memories and interfaces either private to each core or shared among the cores. Some memories and interfaces are optional so that you can configure the memory system according to your system requirements.



Note

- When you choose to exclude the optional ITCM and DTCM, the logic is removed.
- When you choose to exclude the optional L2 memory, the logic is always present but the RAM size is 0.
- When you choose to exclude the optional LLRAM, SPP, and LLPP interfaces, all the associated logic is removed. See [1.3.1 Configuration parameters](#) on page 15 for more information.

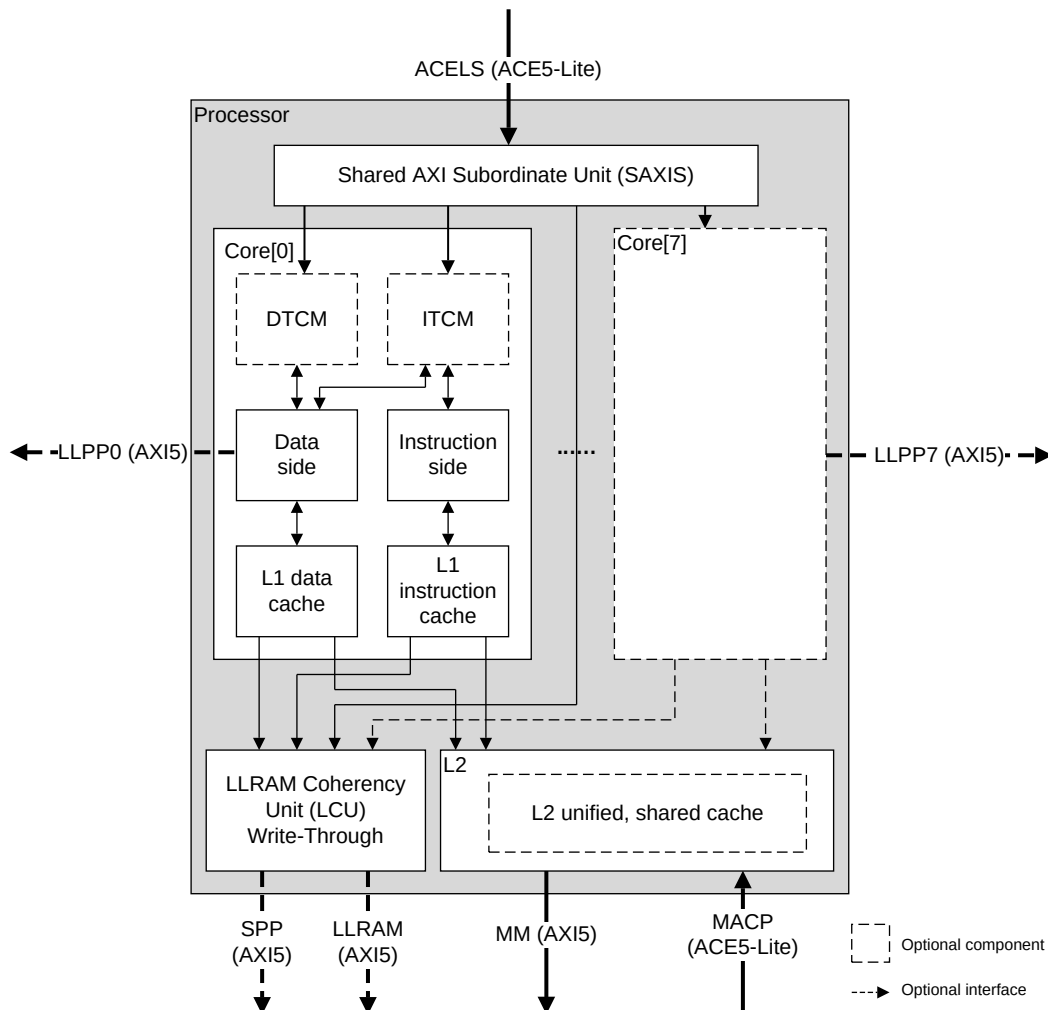
The memory system consists of:

- Memories private to each core:
 - An L1 instruction cache.
 - An L1 data cache.
 - An optional *Instruction Tightly Coupled Memory* (ITCM) for instructions and data.
 - An optional *Data Tightly Coupled Memory* (DTCM) for data.
- An optional, per-core AXI5 32-bit *Low-latency Peripheral Port* (LLPP) manager interface.
- Configurable 40 or 48-bit physical address width.
- An optional, shared, and unified L2 cache.
- A shared *Main Manager* (MM) interface implemented as either CHI.E or AXI5 256-bit port.
- A shared *Main Manager* (MM) interface implemented as AXI5 256-bit port.
- An optional and shared AXI5 64-bit *Shared Peripheral Port* (SPP) manager interface.
- An optional and shared AXI5 256-bit *Low-latency RAM* (LLRAM) manager interface.
- A shared ACE5-Lite 128-bit *Main Accelerator Coherency Port* (MACP) subordinate interface for external access to MM address ranges.
- A shared 128-bit *ACE-Lite Subordinate* (ACELS) interface used for two purposes:

- As an LLRAM Accelerator Coherency Port (ACP) that enables coherent external access to the LLRAM.
- As a TCM subordinate that enables external access such as *Direct Memory Access* (DMA) to the TCMs within the cores.

The following figure shows the Cortex®-R82 processor memory system and external interfaces.

Figure 8-1: Memory system



The TCM, LLPP, SPP, and LLRAM base addresses are configurable through top-level *BASEADDR pins. See [1.3.2 Integration-time configuration options](#) on page 21 for more information on these pins. Software has read-only access to those base addresses through related region registers. See [A.2.2.39 IMP_ITCMREGIONR_EL1, ITCM Region Register](#) on page 499, [A.2.2.40 IMP_DTCMREGIONR_EL1, DTCM Region Register](#) on page 502, [A.2.2.41 IMP_LLPPREGIONR_EL1, LLPP Region Register](#) on page 505, [A.2.2.43 IMP_SPPREGIONR_EL1,](#)

[SPP Region Register](#) on page 510, and [A.2.2.42 IMP_LLDRAMREGIONR_EL1, LLDRAM Region Register](#) on page 508 for more information on these registers.



DED ECC is provided for TCMs, the L1 data cache data RAMs, L1 data cache dirty RAMs, L2 cache tag and data RAMs, L2 cache data buffers, and L2 and LCU duplicates of L1 tag RAMs. DED ECC is provided for the L1 instruction cache data RAMs, L1 instruction cache tag RAMs, and L1 data cache tag RAMs. L2 cache replacement RAM and branch predictor RAMs are not protected.

The Cortex®-R82 processor provides protection against 1-bit and 2-bit errors for all functional RAMs. Where an error can cause a functional impact Cortex®-R82 implements a *Single Error Correct Double Error Detect* (SECCDED) scheme if correction is required or a *Double Error Detect* (DED) scheme otherwise.

The Cortex®-R82 processor memory system provides various memories and interfaces each tailored to different requirements. The aim is that some memories and interfaces are used for more critical real-time requirements and some for less critical real-time requirements. The more real-time critical context is also able to access the less real-time critical interfaces and memories although such an access might not be desirable depending on the system design.

The Cortex®-R82 processor memories and interfaces can be ordered as follows in terms of meeting critical real-time requirements:

1. TCMs are the most deterministic.
2. LLDRAM, cached through the L1 caches, is more deterministic than the MM but less deterministic than the TCMs.
3. MM, cached through the L1 and L2 caches, is the least deterministic.

LLPP is a similar level of determinism to TCMs, and SPP is a similar level of determinism to LLDRAM.

The TCMs provide the most deterministic timing for memory accesses. The ITCM enables low-latency access for instructions and data. The DTCM provides low-latency access for data only. The ITCM and the DTCM are private to each core and their contents are not cached.

The TCM interface includes a full crossbar switch that allows concurrent access from three requesters (instruction side, data side, and ACELS) to the ITCM and concurrent access from two requesters (data side and ACELS) to the DTCM. If one or more requesters attempt to access the same TCM, the TCM interface arbitrates between the requests on a fixed priority scheme with Quality of Service (QoS) mechanisms. See [8.14.4 Quality of Service](#) on page 193 for information on QoS.

The TCMs support memory testing via *Memory Built-In Self Test* (MBIST).

Where access timing determinism is less critical but fast access is still required, L1 instruction cache and L1 data cache can be used. L1 instruction cache and L1 data cache are used to cache instructions and data respectively from the MM port and the LLDRAM port. The cache behavior

depends on the memory attributes. The L1 memories support cache maintenance operations according to the Arm® architecture and memory testing via MBIST.

The LLRAM interface is optimized for deterministic, low-latency access. Accesses from the LLRAM interface can be cached in the L1 cache if the memory attributes are cacheable. These accesses are only cached Write-Through to improve the LLRAM determinism by avoiding waiting for evictions. The LLRAM port is expected to be connected directly to an SRAM controller for optimal performance. Under this assumption, the LLRAM is expected to have better real-time characteristics compared to the MM port. A real-time context is also able to access the MM port, although such an access might not be desirable depending on the system design.

The L2 cache is unified and therefore it can cache both instructions and data. L2 can cache instructions and data only from the MM port but not the LLRAM port. Accesses from the MM port are only cached Write-Back in the L1 and L2 caches to limit the bandwidth to the main memory.

The following table shows which accesses are allowed in the *Protected Memory System Architecture* (PMSA) and *Virtual Memory System Architecture* (VMSA) contexts.

Table 8-1: Accesses in PMSA and VMSA

Memory system	PMSA	VMSA
ITCM and DTCM	Allowed	Not allowed (aborted)
LLRAM	Allowed	Page table walk accesses are not allowed. Generates synchronous External abort on translation table walk or hardware update of translation table. The level depends on the current level of the pagewalk. General accesses are allowed.
MM	Allowed	Allowed
LLPP	Allowed	Allowed
SPP	Allowed	Allowed

8.2 TCM memories

The *Tightly Coupled Memories* (TCMs) have the most deterministic memory access timing. They are expected to store the most critical real-time code and data, such as interrupt handlers and memory stacks.

Each core within the Cortex®-R82 processor has two optional TCMs, *Instruction Tightly Coupled Memory* (ITCM) and *Data Tightly Coupled Memory* (DTCM), implemented as RAMs. Optional here means that the minimum required logic is always present but the size can be OKB.

ITCMs are unified and can be used for both instructions and data. They are typically used for instruction side accesses such as storing critical code and exception handlers, but also available for data side read and write accesses such as accessing literal pools or initializing program code. DTCMs are dedicated for data side accesses only, such as stack operations and inter-thread data sharing.



Both ITCM and DTCM are optimized to allow concurrent accesses from multiple sources. Each TCM is organized in two logical banks. Core instruction-side read accesses (for the ITCM only), core data-side read/write accesses, and external read/write accesses from *ACE-Lite Subordinate* (ACELS) interface can simultaneously access a TCM in the same cycle if the accesses are on different TCM banks. Simultaneous requests to the same bank are arbitrated in a fair way to minimize the core stalls.

You can implement each TCM independently to sizes of 0KB, 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, or 1MB. A size of 0KB indicates that the TCM is not implemented.

All writes to the ITCM that are less than 64 bits wide will always require a read-modify-write procedure. Use 64 bit wide accesses to optimize ITCM writes.

The base address and the size of each TCM and whether it is enabled is visible in the respective TCM region register ([A.2.2.39 IMP_ITCMREGIONR_EL1, ITCM Region Register](#) on page 499 and [A.2.2.40 IMP_DTCMREGIONR_EL1, DTCM Region Register](#) on page 502).

If the CFGITCMEN_m input for core *m* is asserted at reset, the ITCM in that core is enabled. The ITCM base address is set by the CFGITCMBASEADDR_m configuration inputs of core *m*.

The DTCM remains disabled unless a write to IMP_DTCMREGIONR_EL1 enables it. The DTCM base address is set by the CFGDTCMBASEADDR_m configuration inputs of core *m*.

If the TCMs are implemented and not enabled, accesses to the TCM regions generate aborts.

Both TCMs are accessible by the software running at EL1 or EL0 only if it operates in a *Protected Memory System Architecture* (PMSA) context.



TCM accesses are not supported in *Virtual Memory System Architecture* (VMSA) context.

Software running at EL2 can always access the TCMs. Writes to TCM region registers from EL1 can be trapped to the hypervisor, running at EL2, by setting the register bit HCR_EL2.TIDCP=1 or ACTLR_EL2.REGIONS=0.

The TCMs can be accessed for reads and writes by external agents through the *ACE-Lite Subordinate* (ACELS) port, implemented as a ACE5-Lite 128-bit shared subordinate interface.

You can implement each TCM independently with a wait state of 0 to 3 cycles on access, meaning that accesses to that TCM incur that many cycles of latency. If you implement some, but not all TCMs with wait cycles, the performance of the TCMs might differ from each other.

ITCMs provide up to 128-bit per cycle of bandwidth to the instruction side, up to 128-bit per cycle of bandwidth to the data side and up to 128-bit per cycle of bandwidth to the external agent

through the ACELS. They allow for 256-bit per cycle of total bandwidth when two of these sources concurrently access oppositely 128-bit-aligned locations.

DTCMs provide up to 128-bit per cycle of bandwidth to the data side and up to 128-bit per cycle of bandwidth to the external agent through the ACELS. This allows 256-bit per cycle of total bandwidth when the core and an external agent through the ACELS concurrently access oppositely 128-bit-aligned data locations. Instruction side accesses to DTCMs are not allowed.

Load and store exclusive accesses performed to the TCMs are handled by the local exclusive monitor.

TCMs are private to each core without any internal mechanism for one core to access another core's TCMs. However, the Cortex®-R82 processor enables any core within the processor to access all the TCMs, except its own, via an interconnect loopback. For more information on the loopback address mapping, see [8.10.4 TCM subordinate](#) on page 172.

The Cortex®-R82 processor supports optional SECCDED ECC protection with 64-bit chunk for the ITCM and optional SECCDED ECC protection with 32-bit chunk for the DTCM.

For more information on TCM memory protection behavior, see [10.2 Memory protection behavior](#) on page 219.

TCM attributes and permissions

Enabled TCMs always behave as Non-cacheable Non-shareable Normal memory. This is irrespective of the memory type attributes defined in the *Memory Protection Unit* (MPU) for a memory region containing addresses that are held in the TCM. Access permissions for TCM accesses are the same as the permission attributes that the MPU assigns to the same address.

8.3 L1 memory system

Each core within the Cortex®-R82 processor has separate L1 instruction and L1 data caches. You can configure the L1 instruction and data caches within a core independently from each other and also from the L1 caches in other cores during rendering.

The L1 instruction cache size can be configured to 16KB, 32KB, 64KB, or 128KB. The L1 data cache size can be configured to 16KB, 32KB, or 64KB.

L1 caches improve the average performance of programs that do not use the *Tightly Coupled Memories* (TCMs). Because the performance of an individual transaction depends on whether the cache hits or misses, performance from L1 caches is less deterministic than the TCMs.

Both the L1 instruction and L1 data cache can cache data from the *Main Manager* (MM) port and the *Low-latency RAM* (LLRAM) port. The L1 data cache supports Write-Back caching for MM locations and Write-Through caching for LLRAM locations.

The L1 caches fetch critical-word first. Linefills support write streaming and are non-blocking to subsequent cache accesses. The *Memory Protection Unit* (MPU) and the *Memory Management Unit*

(MMU) control the cacheability of accesses. The MPU and MMU also provide allocation hints, although these may be automatically overridden to improve the performance when write streaming is detected. For more information on write streaming see, [8.3.2.4 Write Streaming Mode](#) on page 114.

The Cortex®-R82 processor includes both instruction side and data side prefetchers to improve the cache hit rate, and therefore, the performance. The prefetchers can anticipate several variants of cache access patterns, and request linefills to the L1 caches and also to the shared L2 cache.

The Cortex®-R82 processor does not include any virtually indexed caches that could generate aliasing issues, so cache maintenance is not required on context switches.

The L1 caches are automatically invalidated after a core reset, unless the reset is because of an MBIST request or for debug recovery.

The Cortex®-R82 processor provides an optional *Error Correcting Code (ECC)* scheme for detecting and correcting errors. L1 instruction cache RAMs and L1 data cache tag RAMs are protected with *Double Error Detect (DED)* scheme. L1 data cache data RAMs are protected with *Single Error Correct Double Error Detect (SECCDED)* scheme. ECC codes are generated and checked automatically when ECC is enabled. Errors are corrected when possible. Otherwise an abort is taken by the Cortex®-R82 processor if the error is about to be consumed or a poison code is written to the cache. The caches are tolerant to a limited number of hard errors.



In implementations with core cache RAM protection, DED ECC bits protect the L1 instruction cache data and tag RAMs by enabling the detection of any single or double bit error. If an error is detected, the line is invalidated and fetched again.

For more information on L1 memory protection behavior, see [10.2 Memory protection behavior](#) on page 219.

L1 instruction side memory system

The L1 instruction side memory system provides an instruction stream to the *Data Processing Unit (DPU)*. Its key features are:

- 64-byte instruction side cache line length.
- 4-way set associative L1 instruction cache.
- 128-bit read interface to the shared L2 memory system.
- Instruction side prefetcher that prefetches cache lines ahead of the current point of execution from the MM or LLRAM interfaces and allocates them into the L1 instruction cache.

The Cortex®-R82 processor uses extensive branch prediction to improve *Instructions Per Cycle (IPC)* and power efficiency.

L1 data side memory system

The L1 data side memory system responds to load and store requests from the DPU. It also responds to L2 coherency logic snoop requests from other cores or external agents. Its key features are:

- 64-byte data side cache line length.
- 4-way set associative L1 data cache.
- Read buffer that services both the *Data Cache Unit* (DCU), and the *Instruction Fetch Unit* (IFU).
- 128-bit read path from the data L1 memory system to the datapath.
- 128-bit write path from the datapath to the L1 memory system.
- Merging store buffer capability which writes to all types of memory (Device, Normal Cacheable and Normal Non-cacheable).
- Data side prefetcher that prefetches data cache lines ahead of the current point of execution from the MM or LLRAM interfaces and allocates them into the L1 data cache. Data side prefetcher is capable of detecting both constant and patterns of strides.

The L1 data side memory system includes forwarding paths to reduce load-to-use time where possible. These are only supported when data accesses are made in little-endian format.

8.3.1 L1 instruction memory system

The L1 instruction cache is organised as a *Virtually Indexed Physically Tagged* (VIPT) cache.

The L1 instruction side memory system provides an instruction stream to the *Data Processing Unit* (DPU).

8.3.1.1 Instruction cache disabled behavior

If the SCTLR_EL1.I bit is set to 0, load and store instructions at EL1 and EL0 do not access any of the L1 instruction or L2 caches. If the SCTLR_EL2.I bit is set to 0, load and store instructions at EL2 do not access any of the L1 instruction or L2 caches.

The SCTLR_EL1.I and SCTLR_EL2.I bits control whether accesses from the core can look up and allocate into the L1 instruction cache and unified L2 cache. L1 instruction cache maintenance operations execute normally, regardless of how the SCTLR_EL1.I and SCTLR_EL2.I bits are set.

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

The Cortex®-R82 processor does not allocate lines into the instruction cache when the instruction cache is disabled or the memory is marked as non-cacheable.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

8.3.1.2 Instruction cache speculative memory accesses

Instructions that are fetched may get discarded and in that sense are speculative, as 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. Instruction fetches to Device type memory take a synchronous prefetch abort.

Device memory areas 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.

If the instruction cache is enabled and if the instruction fetches miss in the L1 instruction cache, instruction fetches can still look up and return data from the L2 cache or L1 data cache if appropriate, or from the external interfaces otherwise. Instruction fetches never result in a line being allocated in the L1 data cache but may result in the line being allocated into the L2 cache if the L2 cache is enabled.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

8.3.1.3 Instruction cache invalidate on reset

The Arm®v8-R AArch64 architecture supports the system instructions `IC IALLU` and `IC IALLUIS` to invalidate the entire instruction cache.

The Cortex®-R82 processor automatically invalidates instruction caches on reset unless suppressed with the debug recovery P-Channel state or an MBIST request. It is therefore not necessary for software to invalidate the instruction caches on startup.

8.3.1.4 Branch prediction

The Cortex®-R82 processor contains branch prediction hardware, also known as program flow prediction.

Branch prediction increases overall performance and reduces power consumption. With branch prediction disabled, all taken branches incur a penalty that is associated with flushing the pipeline.

To avoid this penalty, the branch prediction hardware predicts:

- If a conditional or unconditional branch is to be taken.
- The address to which the branch goes, known as the branch target address.

For conditional branches, the hardware predicts if the branch is to be taken and also the branch target address. For unconditional branches, only the branch target address is predicted.

The hardware contains the following functionality:

- A *Branch Target Address Cache* (BTAC) holding the branch target address of previously taken indirect branches.
- A dynamic branch predictor.
- The return stack, a stack of nested subroutine return addresses.
- A static branch predictor.

Predicted and non-predicted instructions

As a rule, the flow prediction hardware predicts all branch instructions regardless of the addressing mode, and includes:

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

Exception return and exception generating instructions are not predicted.

Return stack

On execution of Branch with Link instructions, the return stack stores the address that is equal to the link register value stored in X30.

The following instructions cause a return stack push:

- BL
- BLR
- BLRAA, BLRAAZ, BLRAB, BLRABZ

The return instructions cause a return stack pop. These include:

- RET
- RETAA, RETAB

The exception return instructions (`ERET`, `ERETA`, and `ERETAB`) and exception generating instructions (`SVC`, `HVC`, `BRK`, and `HLT`) are not predicted because they can change the privilege mode. Prefetching stops when any of these instructions are encountered until they are committed. Then the fetch resumes from the appropriate point in the program flow.

Similarly, certain barrier instructions such as `SB` and `ISB` also cause prefetching to stop, to help lower the attack surface of various side-channel attacks.

8.3.1.5 Instruction prefetching

The Cortex®-R82 processor includes an instruction side prefetcher that prefetches cache lines ahead of the current point of execution from the *Main Manager* (MM) or *Low-latency RAM* (LLRAM) interfaces and allocates them into the L1 instruction cache.

The Cortex®-R82 processor has IMP_CPUACTLR_EL1 system register controls for the instruction side prefetcher to:

- Enable or disable the prefetcher with or without power-aware throttling mechanisms.
- Configure how far ahead from the current point of execution to prefetch, that is, 1 to 4 cache lines ahead of the current line.
- Configure whether the prefetcher always prefetches ahead or only starts prefetching ahead on an L1 instruction cache miss.

8.3.2 L1 data memory system

The L1 data cache is organized as a *Physically Indexed Physically Tagged* (PIPT) cache.

8.3.2.1 Data cache disabled behavior

If the SCTLR_EL1.C bit is set to 0, load and store instructions at EL1 and EL0 do not access any of the L1 data or L2 caches. If the SCTLR_EL2.C bit is set to 0, load and store instructions at EL2 do not access any of the L1 data or L2 caches.

The SCTLR_EL1.C and SCTLR_EL2.C bits control whether accesses from the core can look up and allocate into the L1 data cache and unified L2 cache. L1 data cache maintenance operations execute normally, regardless of how the SCTLR_EL1.C and SCTLR_EL2.C bits are set.

If the L1 data and L2 caches are disabled at the current Exception level, then the following applies:

- All load and store instructions to cacheable memory are treated as if they were non-cacheable. Therefore, they are not coherent with the caches in this core or the caches in other cores, and software must take this into account.

The L1 data and L2 caches cannot be disabled independently.

8.3.2.2 Data cache maintenance considerations

The `DC ivac` instruction performs an invalidate of the target address.

If the data is dirty, a clean is performed before the invalidate.

The `DC isw`, `DC csw`, and `DC csw` instructions perform both a clean and invalidate of the target set/way. The value of HCR_EL2.SWIO has no effect and it is implemented as `RES1` in the Cortex®-R82 processor.



There might be implications to real-time characteristics of *Low-latency RAM* (LLRAM) accesses when `DC ISW`, `DC CSW`, or `DC CISW` instructions are executed. See [8.14 Real-time considerations](#) on page 182 for more information.

8.3.2.3 Data cache coherency

The Cortex®-R82 processor uses the MESI protocol to maintain data coherency between multiple cores.

MESI describes the state that a shareable line in a L1 data cache can be in:

M	Modified/ <i>UniqueDirty</i> (UD). The line is in only this cache and is dirty.
E	Exclusive/ <i>UniqueClean</i> (UC). The line is in only this cache and is clean.
S	Shared/ <i>SharedClean</i> (SC). The line is possibly in more than one cache and is clean.
I	Invalid/ <i>Invalid</i> (I). The line is not in this cache.

The *Data Cache Unit* (DCU), the L2, and the *LLRAM Coherency Unit* (LCU) store the MESI state of the cache lines in the tag, dirty, and the L1 duplicate tag RAMs.



The names *UniqueDirty*, *SharedDirty*, *UniqueClean*, *SharedClean*, and *Invalid* are the AMBA names for the cache states. The Cortex®-R82 processor does not use the *SharedDirty* AMBA state.

8.3.2.4 Write Streaming Mode

A cache line is allocated to the L1 cache on either a read miss or a write miss.

However, there are some situations where allocating on writes is not desirable. For example, when executing the C standard library `memset()` function to clear a large block of memory to a known value. Writes of large blocks of data can pollute the cache with unnecessary data. It can also waste power and performance if a linefill must be performed only to discard the linefill data because the entire line was subsequently written by the `memset()`.

To counter this, the *Store Unit* (STU) includes logic to detect when the core has written at least a byte in all the four chunks of the cacheline, before the linefill completes. The granularity of a chunk is 128 bits. The granularity of a cacheline is 512 bits.

If this situation is detected on a configurable number of consecutive linefills, then it switches into write streaming mode. This is sometimes referred to as read allocate mode.

When in write streaming mode:

- Loads behave as normal and can still cause linefills.

- Writes still lookup in the cache but if they miss then, they write out to L2 cache rather than starting a linefill.



More than the specified number of linefills might be observed on the AXI or CHI manager interface, before the STU detects that the programmed number of full cache lines have been written and switches to write streaming mode.

The STU continues in write streaming mode until it detects either a cacheable write burst that is not a full cache line, or there is a load from the same line as is currently being stored to.

In the Cortex®-R82 processor:

- IMP_CPUACTLR_EL1.DL1WS configures the L1 write streaming mode threshold.
- IMP_CPUACTLR_EL1.DL2WS configures the L2 write streaming mode threshold.

8.3.2.5 Data cache invalidate on reset

The Arm®v8-R AArch64 architecture does not support an operation to invalidate the entire data cache.

The Cortex®-R82 processor automatically invalidates data caches on reset unless suppressed with the debug recovery P-Channel state or an MBIST request. It is therefore not necessary for software to invalidate the data caches on startup.

If software requires this function later after the startup, it must be constructed by iterating over the cache geometry and executing a series of individual invalidate by set/way instructions.

8.3.2.6 Instructions implemented by the L1 memory system

This section describes the instructions implemented by the L1 memory system.

Atomic instructions

The Cortex®-R82 processor supports the atomic instructions added in the Arm®v8.1 architecture.

Accesses targeting the *Main Manager* (MM) interface perform all the atomics either in the L1 data cache or in the interconnect outside the Cortex®-R82 processor. MM 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:

- Cacheable MM atomics are executed near if BROADCASTATOMICM is LOW.
- Non-cacheable/Device LLRAM atomics are executed far when CFGLLRAMSHARED is HIGH and BROADCASTATOMICL is high. When CFGLLRAMSHARED is HIGH and BROADCASTATOMICL is LOW, they are aborted by the Cortex®-R82 processor. When CFGLLRAMSHARED is LOW, they are executed near.

- Cacheable MM atomics are executed near if the shareability domain does not extend to the interconnect (BROADCASTOUTERM is LOW for shareable atomics).
- Cacheable MM atomics are executed near if they hit in the cache in a unique state.
- Unaligned cacheable MM atomics are always executed near.
- Unaligned atomics on the *Tightly Coupled Memories* (TCMs) are always executed near.
- Unaligned atomics to *Low-latency RAM* (LLRAM) port, *Shared Peripheral Port* (SPP), and *Low-latency Peripheral Port* (LLPP) are not supported.
- Non-cacheable/Device MM atomics are always executed far, unless BROADCASTATOMICM is LOW in which case they are aborted by the Cortex®-R82 processor.
- Unaligned atomics to LLRAM port are not supported.
- Atomics to SPP and LLPP are not supported.
- Cacheable MM atomics which miss in the cache are executed far if BROADCASTATOMICM is HIGH.
- Cacheable MM atomics which hit in the cache in a shared state execute far if BROADCASTATOMICM is HIGH.

The Cortex®-R82 processor MM supports sending atomics externally to Device or Non-cacheable memory, however this depends on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them (BROADCASTATOMICM is LOW), it results in a synchronous Data Abort for both load atomics and store atomics. If there is an external abort on an atomic, it results in a synchronous Data abort for load atomics and an asynchronous Data abort for store atomics.

The behavior of the atomic instructions can be modified by the IMP_CPUACTLR_EL1 register settings. For more information on the IMP_CPUACTLR_EL1 register, see [A.2.2.44 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register](#) on page 512.

Low-latency RAM (LLRAM) manager interface performs all aligned atomics within the *LLRAM Coherency Unit* (LCU) at the shared level rather than in the L1 memory system or external memory. The Cortex®-R82 processor LLRAM performs atomics to device or Non-cacheable memory within the LCU without requiring interconnect support. This is built on the assumption that the LLRAM port is connected directly to an external memory and the LLRAM manager cannot access data that is shared with other agents in the system. The Cortex®-R82 processor can perform atomics on the LLRAM port assuming that no external agents are able to modify the memory location between the read and write operation, and caches shareable data without any support for cache coherency with external agents. This only applies when CFGLLRAMSHARED is set to 0 or when CFGLLRAMSHARED is set to 1 but for non-shareable/inner-shareable locations.

LDAPR instructions

The Cortex®-R82 processor supports load-acquire and store-release instructions with the RCpc consistency semantic introduced in the Armv8.4-RCpc extension, however the implementation does not make use of the ordering relaxations. Armv8.4-RCpc load-acquire and store-release instructions will still be ordered when targeting different addresses.

The instruction support is reflected in register ID_AA64ISAR1_EL1 where bits[23:20] are set to 0b0010 to indicate that the processor supports `LDAPUR*`, `STLUR*`, and `LDAPR*` instructions.

For more information on the ID_AA64ISAR1_EL1 register, see [A.2.1.33 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1](#) on page 354.

Transient memory region

The Cortex®-R82 processor has a specific behavior for memory regions that are marked as Write-Back cacheable and transient for MM and Write-Back transient or Write-Through transient for LLRAM, as defined in the Arm®v8-R AArch64 architecture.

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

- If the memory access misses in the L1 data cache, the returned cache line is allocated in the L1 data cache but is marked as transient. Where an L1 data cache allocation needs to evict an existing entry, transient locations are prioritized for eviction where possible.

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

- On eviction, if the line is clean and marked as transient, it is not allocated into the L2 cache but is marked as invalid.

For MM, for stores that are targeted at a memory region that is marked as transient, the following occurs:

- If the store misses in the L1 data cache, the line is allocated into the L2 cache.

Non-temporal loads

Non-temporal loads indicate to the caches that the data is likely to be used for only short periods. For example, when streaming single-use read data that is then discarded. In addition to non-temporal loads, there are also prefetch-memory (`PRFM`) hint instructions with the `STRM` qualifier.

Non-temporal loads cause allocation into the L1 data cache, with the same performance as normal loads. However, when a later linefill is allocated into the cache, the cache line marked as non-temporal has higher priority to be replaced. To prevent pollution of the L2 cache, a non-temporal line that is evicted from L1, is not allocated to L2 as would happen for a normal line.



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

Non-temporal stores are treated the same as stores to a memory region that is marked as transient.

8.3.2.7 Local exclusive monitor

The Cortex®-R82 processor L1 memory system has an architecturally defined local exclusive monitor.

This monitor is a 2-state, open and exclusive, state machine that manages Load-Exclusive or 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.ERG` defines the size of the tagged block as 16 words, one cache line.



A load/store exclusive instruction is any instruction that has a mnemonic starting with `LDX`, `LDAX`, `STX`, or `STLX`.

If a Load-Exclusive instruction is performed to non-cacheable shareable or device memory and is to a region of memory in the SoC that does not support exclusive accesses, the external exclusive-fail response causes a Data Abort exception with a Data Fault Status Code of `0b110101`.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information about these instructions.

8.3.2.7.1 Treatment of intervening STR operations

Where there is an intervening store operation between an exclusive load and an exclusive store from the same core, the intermediate store does not produce any direct effect on the local exclusive monitor.

After the exclusive load, the local monitor is in the Exclusive Access state. It remains in the Exclusive Access state after the store, and then returns to the Open Access state only after an exclusive store, a `CLREX` instruction, or an exception return.

However, if the exclusive code sequence accessed address is in cacheable memory, any eviction of the cache line containing that address clears the monitor. Arm recommends that no load or store instructions are placed between the exclusive load and the exclusive store, because these additional instructions can cause a cache eviction. Any data cache maintenance instruction can also clear the exclusive monitor.

8.3.2.7.2 Exclusive monitor

In the exclusive state machine, the **IMPLEMENTATION DEFINED** transitions are as follows:

- If the monitor is in the exclusive state, and a store exclusive is performed to a different cacheline, then the store exclusive fails and does not update memory.

- If the monitor is in the exclusive state and a store exclusive is performed to a same cacheline but a different address, then the store exclusive passes and updates memory.
- If a normal store is performed to a different address, it does not affect the exclusive monitor.
- If a normal store is performed from a different core to the same address for Sharable Cacheable locations, it clears the exclusive monitor. If the store is from the same core then it does not clear the monitor.
- If a normal store is performed from a different core to the same address for Sharable Non-cacheable locations, then the global exclusive monitor in the SoC clears the exclusive monitor. If the store is from the same core then it does not clear the monitor.

**Note**

If you are using load/store exclusive instructions to build semaphores, Arm recommends that you use a full cacheline per semaphore.

8.3.2.8 Data prefetching

The following section describes the software and hardware data prefetching behavior of the Cortex®-R82 processor.

Hardware data prefetcher

The Cortex®-R82 processor has a data prefetch mechanism that looks for cache line fetches with regular patterns. If the data prefetcher detects a pattern, then it signals to the memory system that memory accesses from a specified address are likely to occur soon. The memory system responds by starting new linefills to fetch the predicted addresses ahead of the instruction stream.

The Cortex®-R82 processor can track multiple streams in parallel.

Prefetch streams end when either:

- The pattern is broken.
- A `DSB` instruction is executed.
- A `WFI` instruction or `WFE` instruction is executed.

For read streams, the prefetcher is based on the addresses. A given stream is allowed to prefetch addresses through multiple *Memory Protection Unit* (MPU) regions as long as they are cacheable and with read permissions. For more information see, [9.2.1 MPU regions](#) on page 197.

For some types of pattern, when the prefetcher is confident in the stream, it can start progressively increasing the prefetch distance ahead of the current accesses. These accesses start to allocate to the L2 cache rather than L1. Allocating to the L2 cache allows better utilization of the larger resources available at L2. Also, utilizing the L2 cache reduces the amount of pollution of the L1 cache if the stream ends or is incorrectly predicted. If the prefetching to L2 was accurate, the line will be removed from L2 and allocated to L1 when the stream reaches that address.

The IMP_CPUACTLR_EL1 register allows you to:

- Enable or disable the prefetcher, for each target interface separately.
- Control the maximum number of prefetch streams that can be active at once.

Preload instructions

The Cortex®-R82 processor supports `PLD` and `PRFM` instructions. `PLD` and `PRFM` instructions perform a lookup in the L1 cache and, in the case of a cache miss, initiate a linefill request to bring the line into the L1 cache. The `PRFM` instruction also enables targeting of a prefetch to the L2 cache. A request is sent to L2 to start a linefill, and then the instruction can retire without any data being returned to L1. `PLI`, `PLIL1KEEP` and `PLIL1STRM` are implemented as a prefetch to L2.

Use the `PLD` or `PRFM` instruction for software data prefetching where short sequences or irregular pattern fetches are required. For more information about prefetch memory and preloading caches, see the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

Data Cache Zero

The Data Cache Zero by Virtual Address (`DC ZVA`) instruction enables a block of 64-bytes in memory, which is aligned to 64-bytes in size, to be set to 0. The `DCZID_ELO` register passes this value.

The `DC ZVA` instruction allocates this value into the data cache using the same method as a normal store instruction.

8.4 L2 memory system

The Cortex®-R82 processor has an optional unified L2 cache that is shared by all the cores in the cluster. L2 cache improves the average performance of programs that use memory within the *Main Manager* (MM) address range.

The L2 memory is unified and therefore it can cache both instructions and data. The L2 can cache instructions and data only from the MM port but not the *Low-latency RAM* (LLRAM) port. The L2 cache supports Write-Back caching for MM locations.

Because the performance of an individual transaction depends on whether the L1 and L2 caches hit or miss, performance from the L2 cache is less deterministic than the *Tightly Coupled Memories* (TCMs) or LLRAM. The L2 cache fetches critical-word first, and linefills support streaming. This prevents the blocking of subsequent cache accesses.

The *Memory Protection Unit* (MPU) and the *Memory Management Unit* (MMU) control the cacheability of accesses. They also provide allocation hints although these may be automatically overridden to improve the performance when streaming is detected.

The L2 memory subsystem consists of:

- An optional 8-way, set-associative L2 cache with a configurable size of 0KB, 96KB, 128KB, 192KB, 256KB, 384KB, 512KB, 768KB, 1MB, 1.5MB, 2MB, 3MB, or 4MB. Cache lines have a fixed length of 64 bytes.
- Optional *Error Correcting Code* (ECC) protection for tag, data, and L2 data buffer RAM structures.



For information on L2 memory protection behavior, see [10.2 Memory protection behavior](#) on page 219.

The main features of the L2 memory system are:

- Shared and unified L2 cache.
- Organized as a *Physically Indexed Physically Tagged* (PIPT) cache.
- Duplicate PIPT L1 tag RAMs.
- Pseudo-exclusive with L1 data cache.
- Pseudo-inclusive with L1 instruction cache.
- Configurable cache slice and RAM partitions that the L2 cache implements.

The Cortex®-R82 processor respects cacheability attributes when determining whether or not to cache data in the L2 cache.

The L2 cache supports stashing requests from the *Main Accelerator Coherency Port* (MACP) implemented as ACE-Lite and the MM if it is configured as a CHI port.

The L2 cache is invalidated automatically at reset unless the reset is because of an MBIST request or for debug recovery.

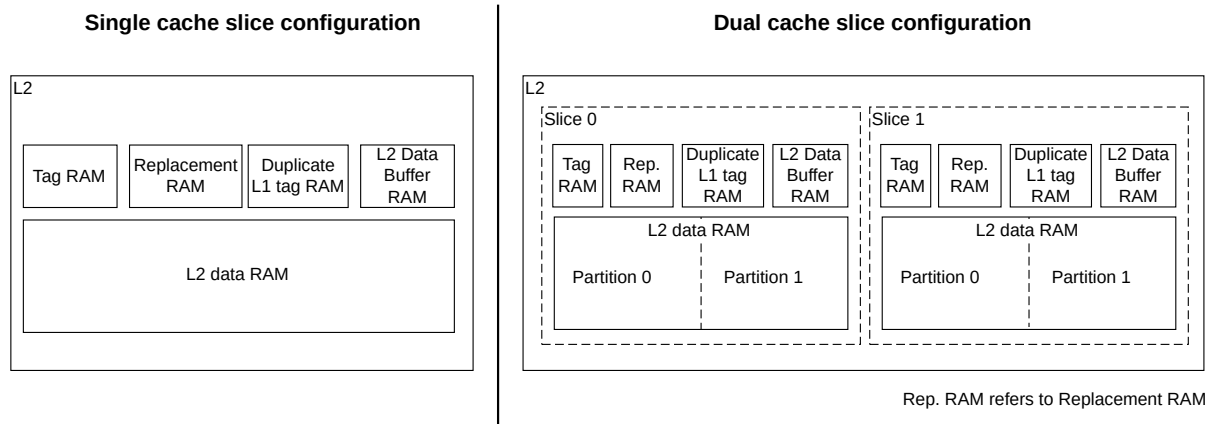
8.4.1 L2 cache slice integration

The Cortex®-R82 processor L2 cache is implemented as either a single cache slice or a dual cache slice, depending on the configuration.

`L2_SLICES` parameter controls the number of slices and RAM partitions the L2 cache implements.

A cache slice consists of data RAMs, tag RAMs, replacement RAMs, duplicate L1 tag RAMs, L2 data buffer RAMs, and associated logic.

When two cache slices are implemented, the overall cache is divided across the two slices. There is associated logic for each of the cache slices. The following figure shows the differences between a single slice configuration and a dual slice configuration.

Figure 8-2: Comparison between a single and dual L2 cache slice configuration

Splitting the cache into two slices improves the physical floorplan when implementing the macrocell, particularly for larger cache sizes. It also gives increased bandwidth because the two slices can be accessed in parallel.

When a dual slice configuration is implemented, the L2 cache data RAM is further divided into two partitions in order to facilitate powering down of half the data RAM when bandwidth requirements are low.

8.4.1.1 Cache slice selection

For a dual cache slice implementation, requests are sent to a particular slice depending on the address and the memory attributes of the request:

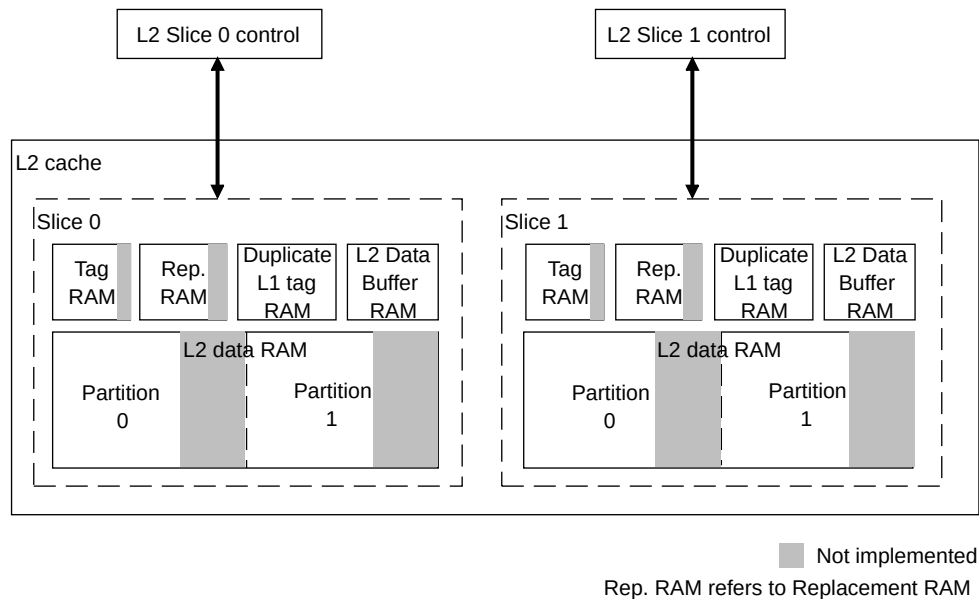
- For Cacheable and Non-cacheable requests, addresses are interleaved between slice 0 and slice 1, based on address bit 6 of the request.
- Device and DVM requests are always sent to slice 0.

8.4.1.2 Non-power-of-two L2 cache implementation

The Cortex®-R82 processor supports the following non power-of-two L2 cache sizes: 96KB, 192KB, 384KB, 768KB, 1.5MB, 3MB.

When configured with a non power-of-two size, the number of L2 cache ways remains unchanged at 8 ways, but the overall number of sets is reduced compared to the equivalent power-of-two configuration.

The following figure shows a non power-of-two L2 cache implementation, highlighting the reduction in available sets across all slices and partitions when compared to the equivalent power-of-two size.

Figure 8-3: Non-power-of-two L2 cache implementation

8.4.2 L2 cache allocation policy

The L2 cache data allocation policy changes depending on the pattern of data usage.

Exclusive allocation is used when data is allocated in only one core. Inclusive allocation is used when data is shared between cores.

For example, an initial request from core 0 allocates data in the L1 caches but is not allocated in the L2 cache. When data is evicted from core 0, the evicted data is allocated in the L2 cache. The allocation policy of this cache line is still exclusive. If core 0 refetches the line, it is allocated in the L1 caches of core 0 and removed from the L2 cache, keeping the line exclusive. If core 1 then accesses the line for reading, it remains cached in core 0 and is also allocated in both core 1 and L2 caches. In this case, the line has inclusive allocation.

8.4.3 L2 cache partitioning

The L2 cache supports a partitioning scheme that alters the victim selection policy to prevent one core (or a group of cores) from using the entire cache at the expense of another core.

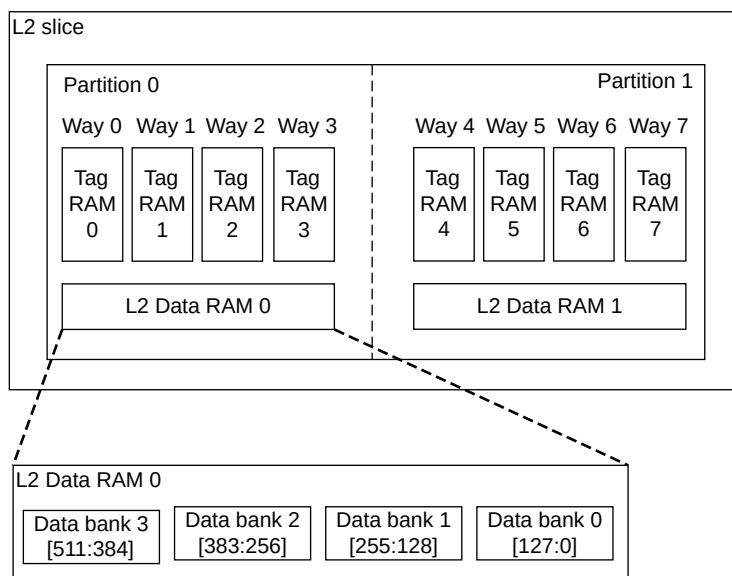
Cache partitioning is intended for specialized software where there are distinct classes of processes running with different cache accessing patterns. For example, two processes A and B run on

separate cores in the same cluster and therefore share the L2 cache. If process A is more data-intensive than process B, then process A can cause all the cache lines that process B allocates to be evicted. Evicting these allocated cache lines can reduce the performance of process B.

To avoid this, the Cortex®-R82 processor supports L2 cache partitioning to enable differentiation between groups of cores with varying types of workload.

The L2 cache implements eight ways. When configured as a single slice, these all fall within a single partition, whereas a dual slice configuration implements four ways per partition as shown in the following figure.

Figure 8-4: L2 cache slice structure



To implement the cache way partitioning, the Cortex®-R82 processor groups pairs of ways into Way Groups and associates these with the different cores via Scheme IDs.

The eight cache ways are arranged into groups as follows:

- Way Group 0: Way 0 and Way 4
- Way Group 1: Way 1 and Way 5
- Way Group 2: Way 2 and Way 6
- Way Group 3: Way 3 and Way 7

Eight combinations of these Way Groups can then be defined via the Cluster Partition Control Register, IMP_CLUSTERPARTCR_EL1. Each grouping is known as a Scheme ID. The Cluster Partition Control Register can be programmed as follows:

- Each way group can be assigned as private to one or more scheme IDs.
- Each way group can be left unassigned and therefore shared between all scheme IDs.

- Multiple way groups can be assigned as private to a single scheme ID.

These Scheme IDs can then be configured for use as follows:

- Each core is assigned to a scheme ID with the Cluster Scheme ID register, IMP_CLUSTERSID_EL1 which is banked per-core.
- *Main Accelerator Coherency Port* (MACP) transactions are assigned to a scheme ID with the Cluster MACP Scheme ID register IMP_CLUSTERACPSID_EL1.

Accesses from a given core or the MACP can allocate into any cache way that is assigned as private to the allocated Scheme ID, or to any cache way that is shared between all Scheme IDs.



If some cache ways are powered down, the number of ways in each partition are reduced. If there are insufficient ways available to a process, performance might be reduced. Therefore Arm recommends that using cache partitioning and cache way powerdown at the same time is done with care.

8.4.4 L2 cache stashing

Cache stashing is the ability of an external agent to request that a line is brought in (or stashed) to a cache in the cluster.

Cache stashing can be performed over the *Main Accelerator Coherency Port* (MACP) or the *Main Manager* (MM) interface if it is configured as CHI. All stash requests target only the L2 cache.

Cache stashing is not supported on the *ACE-Lite Subordinate* (ACELS) interface or the MM interface if it is configured as AXI.

8.4.5 L2 cache data RAM latency

The L2 cache data RAM input and output latency is configurable in order to meet a variety of potential frequency targets.

For more information on the L2 cache data RAM latency, see the *Arm® Cortex®-R82 Processor Configuration and Integration Manual*.



The *Arm® Cortex®-R82 Processor Configuration and Integration Manual* is a confidential document that is only available to Cortex®-R82 processor IP licensees.

8.5 LLPP manager interface

The *Low-latency Peripheral Port* (LLPP) provides minimum latency access to memory and devices outside the cluster. Each core within the Cortex®-R82 processor has an optional AXI5 32-bit LLPP manager interface.

The LLPP is expected to be connected to a limited number of latency-critical peripherals that are private to a core. No other core can access to the peripherals connected to a core's LLPP. Similarly, the *ACE-Lite Subordinate* (ACELS) interface cannot access to the LLPP.



You can also connect the LLPP to a wider, shared interconnect, but its latency advantages would be reduced.

You can connect the LLPP to memory instead of peripherals although this is not optimal. This is because the LLPP always applies Device non-Gathering, non-Reordering, with no Early Write Acknowledgement (Device-nGnRnE) memory type and attributes. This means that the Cortex®-R82 processor never speculatively accesses the LLPP, reorders accesses to the LLPP, forwards data from the LLPP, or merges writes to the LLPP.

Any accesses to the LLPP must follow the rules of the Device memory. The LLPP treats Normal memory accesses as if they are Device memory accesses.



If you connect an ideal memory system to the LLPP, then the core can sustain one store word transaction (generated by basic STR instructions) each cycle to the LLPP indefinitely resulting a throughput of 32-bit of store data per cycle.

You can connect the LLPP to AXI4 subordinates, AXI5 subordinates, and AXI5-Lite subordinates.

The LLPP does not support exclusive or atomic transactions. Any exclusive or atomic accesses to the LLPP cause a synchronous External Data Abort that is generated internally rather than requiring the SoC to do so.

Unlike *Main Manager* (MM) accesses, the LLPP accesses are never shared or merged with accesses from a different context, even for Normal type memory. Therefore, it is always possible for the subordinate device to use the *Virtual Machine Identifier* (VMID) signaled on the LLPP to restrict accesses to a particular context.

The LLPP signals the VMID of the context that initiated a transaction on dedicated AXI User signals for all transactions. The LLPP supports only single-beat burst INCR transactions.

The IMP_LLPPREGIONR_EL1 System register holds the information about:

- Whether the LLPP is implemented or not.
- LLPP region size which is fixed to 128MB if the LLPP is implemented and 0 otherwise.

- The LLPP base address.
- Whether the LLPP is enabled or not.

LLPP configuration parameter controls whether the LLPP interface is implemented or not. If the system does not implement the LLPP interface ($LLPP = 0$), the cores in the Cortex®-R82 processor do not include the logic to support LLPP regions and ports. All related AXI signals, CFGLLPPIMP, and CFGLLPPBASEADDR pins are rendered out by the configuration script.

If the system implements the LLPP interface ($LLPP = 1$), all cores in the Cortex®-R82 processor include logic to support the LLPP regions and ports. The LLPP interfaces can be enabled or disabled by the CFGLLPPIMP pin. If the LLPP interface is implemented, the LLPP region has a fixed size of 128MB.

The LLPP base address is set via the configuration signal CFGLLPPBASEADDR and is the same for all cores in a cluster.

The LLPP is disabled at reset. IMP_LLPPREGIONR_EL1.ENABLEEL2 controls whether the LLPP is enabled for access at EL2 and IMP_LLPPREGIONR_EL1.ENABLEEL10 controls whether the LLPP is enabled for access at EL0 and EL1.

For more information about the IMP_LLPPREGIONR_EL1, see [A.2.2.41 IMP_LLPPREGIONR_EL1, LLPP Region Register](#) on page 505.

If the LLPP is implemented and enabled, data accesses to an address in the LLPP region are performed through the LLPP. If the LLPP is implemented and not enabled, data accesses to the LLPP region generate a synchronous External abort. If the LLPP is not implemented, then data accesses to the LLPP region are performed through other parts of the memory system.

If the LLPP is implemented, instruction accesses to an address in the LLPP region cause a synchronous External abort. If the LLPP is not implemented, instruction accesses are performed through other parts of the memory system.

The following table shows the LLPP attributes.

Table 8-2: LLPP attributes

Attribute	Value	Comments
Write issuing capability	8	Each core can issue a maximum of eight write requests.
Read issuing capability	4	Each core can issue a maximum of four LLPP read requests.
Combined issuing capability	12	Each core can issue a maximum of 12 requests.
Exclusive thread capability	0	Exclusive accesses are not supported.
Write ID capability	1	All writes use the same ID.
Write ID width	1	-
Read ID capability	1	All reads use the same ID.
Read ID width	1	-

8.5.1 LLPP features

The following table shows the *Low-latency Peripheral Port* (LLPP) AXI properties the Cortex®-R82 processor supports or requires the cluster interconnect and system to support.

Table 8-3: LLPP features

AXI property	Supported by the Cortex®-R82 processor LLPP	Interconnect support required
Multi_Copy_Atomicity	Yes	Yes
Ordered_Write_Observation	Yes	Yes
Atomic_Transactions	No	No
Poison	No	No
Check_Type	No	No
QoS_Accept	No	No
Trace_Signals	No	No
Loopback_Signals	No	No
Wakeup_Signals	Yes	Yes
Untranslated_Transactions	No	No
NSAccess_Identifiers	No	No

8.5.2 LLPP memory attributes

The LLPP uses the ARCACHEPm and AWCACHEPm signals to indicate the memory attributes of transactions.

The following table shows the encoding that is used for the ARCACHEPm and AWCACHEPm signals of the LLPP manager interface. Because the LLPP is optimized for peripherals, all accesses propagate Device Non-bufferable attributes, regardless of the attributes that are returned by the memory management system translation. All other encodings are unused.

Table 8-4: ARCACHEPm and AWCACHEPm encodings

Encoding	Meaning
0b0000	Device Non-bufferable

8.5.3 LLPP transfers

The LLPP only issues INCR bursts of length one and transfers of up to 32 bits per transfer.

The LLPP read-channel can post:

- Two 32-bit outstanding transactions when a single instruction requests 64 bits of data from a 64-bit aligned location.
- Four 32-bit outstanding transactions when a single instruction requests 128 bits of data from a 128-bit aligned location.

The LLPP does not post two read transactions on the bus for two separate instructions or for data within two separate 64-bit aligned locations.

The LLPP write-channel generates up to eight 32-bit transactions on the bus from one or more instructions. All transactions use the same ID to ensure that ordering is maintained.

8.5.4 LLPP AXI transfer restrictions

The LLPP conforms to the AMBA® 5 AXI specification, but it does not generate all the AXI transaction types that the specification permits.

This section describes the types of AXI transactions that the LLPP generates. If you are designing an AXI Subordinate interface to work only with the Cortex®-R82 LLPP, you can take advantage of these restrictions and the interface attributes to simplify the subordinate.

This section also contains tables that show some examples of the types of AXI burst that the Cortex®-R82 processor generates. However, because a particular type of transaction is not shown here does not mean that the Cortex®-R82 processor does not generate such a transaction.

You can connect the LLPP to both AXI4 subordinates and AXI5-Lite subordinates in addition to AXI5 subordinates.

An AXI4 subordinate device that is connected to the LLPP must be capable of handling every kind of transaction that the AXI4 specification permits, except where there is an explicit statement in this chapter that such a transaction is not generated. You must not infer any additional restrictions from the example tables given.

An AXI5-Lite subordinate device that is connected to the LLPP must be capable of handling every kind of transaction that the AXI5-Lite specification permits, except where there is an explicit statement in this chapter that such a transaction is not generated. You must not infer any additional restrictions from the example tables given.

Restrictions on AXI peripheral transfers describes restrictions on the type of transfers that the LLPP generates. If a core exists and is powered up, BREADYPm and RREADYPm are always asserted. You must not make any assumptions about the AXI handshaking signals, except that they conform to the AXI5 specification.

The LLPP applies the following restrictions to the AXI transactions it generates:

- A burst never transfers more than four bytes.
- The burst length is never more than one transfer.
- No transaction ever crosses a 4-byte boundary in memory.
- All transactions are incrementing (INCR) bursts.
- Transactions to Device memory are always to addresses that are aligned to the transfer size.
- No exclusive accesses are generated.
- No atomic accesses are generated.

8.5.4.1 LLPP Device transactions

This section details LLPP Device transactions.

The following table shows the values of ARADDRPm, ARBURSTPm, ARSIZEPm, and ARLENPm for `LDRB` from bytes 0-3 in Device memory.

Table 8-5: `LDRB` transfers

Address[1:0]	ARADDRPm[7:0]	ARBURSTPm	ARSIZEPm	ARLENPm
0x0 (byte 0)	0x00	INCR	8-bit	1 data transfer
0x1 (byte 1)	0x01	INCR	8-bit	1 data transfer
0x2 (byte 2)	0x02	INCR	8-bit	1 data transfer
0x3 (byte 3)	0x03	INCR	8-bit	1 data transfer

The following table shows the values of ARADDRPm, ARBURSTPm, ARSIZEPm, and ARLENPm for `LDRH` from halfwords 0-1 in Device memory.

Table 8-6: `LDRH` transfers

Address[1:0]	ARADDRPm	ARBURSTPm	ARSIZEPm	ARLENPm
0x0 (halfword 0)	0x00	INCR	16-bit	1 data transfer
0x2 (halfword 1)	0x02	INCR	16-bit	1 data transfer

The following table shows the values of ARADDRPm, ARBURSTPm, ARSIZEPm, and ARLENPm for an `LDR` that transfers a single word, `LDR <w>`, in Device memory.

Table 8-7: `LDR <w>` transfers

Address[1:0]	ARADDRPm	AWBURSTPm	ARSIZEPm	ARLENPm
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer

The following table shows the values of ARADDRPm, ARBURSTPm, ARSIZEPm, and ARLENPm for an `LDR` that transfers two 64-bit registers, an `LDR <x>`, in Device memory.

All accesses using `LDR <x>` instructions to Device memory occur as 32-bit transactions.

Table 8-8: `LDR <x>` transfers

Address[3:0]	ARADDRPm	ARBURSTPm	ARSIZEPm	ARLENPm
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer

The following table shows the values of ARADDRPm, ARBURSTPm, ARSIZEPm, and ARLENPm for an `LDP` from words 0-1-2-3, in Device memory.

Table 8-9: LDP transfers

Address[3:0]	ARADDRPm	ARBURSTPm	ARSIZEPm	ARLENPm
0x0 (word 0)	0x00	INCR	8-bit	1 data transfer
0x4 (word 1)	0x04	INCR	8-bit	1 data transfer
0x8 (word 2)	0x08	INCR	8-bit	1 data transfer
0xC (word 3)	0x0C	INCR	8-bit	1 data transfer

The following table shows the values of AWADDRPm, AWBURSTPm, AWSIZEPm, AWLENPm, and WSTRBPm for an STRB from bytes 0-3 in Device memory.

Table 8-10: STRB transfers

Address[1:0]	AWADDRPm	AWBURSTPm	AWSIZEPm	AWLENPm	WSTRBPm
0x0 (byte 0)	0x00	INCR	32-bit	1 data transfer	0b0001
0x1 (byte 1)	0x00	INCR	32-bit	1 data transfer	0b0010
0x2 (byte 2)	0x00	INCR	32-bit	1 data transfer	0b0100
0x3 (byte 3)	0x00	INCR	32-bit	1 data transfer	0b1000

The following table shows the values of AWADDRPm, AWBURSTPm, AWSIZEPm, AWLENPm, and WSTRBPm for an STRH from halfwords 0-1 in Device memory.

Table 8-11: STRH transfers

Address[1:0]	AWADDRPm	AWBURSTPm	AWSIZEPm	AWLENPm	WSTRBPm
0x0 (halfword 0)	0x00	INCR	32-bit	1 data transfer	0b0011
0x2 (halfword 1)	0x00	INCR	32-bit	1 data transfer	0b1100

The following table shows the values of AWADDRPm, AWBURSTPm, AWSIZEPm, AWLENPm, and WSTRBPm for an STR that writes a single word, STR <wt>, to Device memory.

Table 8-12: STR <wt> transfers

Address[1:0]	AWADDRPm	AWBURSTPm	AWSIZEPm	AWLENPm	WSTRBPm
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer	0b1111

The following table shows the values of AWADDRPm, AWBURSTPm, AWSIZEPm, AWLENPm, and WSTRBPm for an STR that transfers a 64-bit registers, an STR <xt>, to Device memory.

Table 8-13: STR <xt> transfers

Address[3:0]	AWADDRPm	AWBURSTPm	AWSIZEPm	AWLENPm	WSTRBPm
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer	0b1111
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer	0b1111

The following table shows the values of AWADDRPm, AWBURSTPm, AWSIZEPm, AWLENPm, and WSTRBPm for an STR that writes two 64-bit registers, an STR <xt1>, <xt2>, to Device memory.

Table 8-14: STP transfers

Address[3:0]	AWADDRPm	AWBURSTPm	AWSIZEPm	AWLENPm	WSTRBPm
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer	0b1111
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer	0b1111
0x8 (word 2)	0x08	INCR	32-bit	1 data transfer	0b1111
0xC (word 3)	0x0C	INCR	32-bit	1 data transfer	0b1111

**Note**

- A load of a halfword from Device memory addresses 0x1 or 0x3 generates an alignment fault.
- A load of a word from Device memory addresses 0x1, 0x2, or 0x3 generates an alignment fault.
- A load-pair from Device memory addresses 0x1, 0x2, 0x3, 0x5, 0x6, or 0x7 generates an alignment fault.
- A store of a halfword from Device memory addresses 0x1, or 0x3 generates an alignment fault.
- A store of a word to Device memory addresses 0x1, 0x2, or 0x3 generates an alignment fault.
- A store-pair to Device memory address 0x1, 0x2, 0x3, 0x5, 0x6, or 0x7 generates an alignment fault.

8.6 SPP manager interface

The *Shared Peripheral Port* (SPP) provides minimum latency access to memory and devices outside the cluster. The Cortex®-R82 processor has an optional AXI5 64-bit SPP manager interface that is shared among the cores within the cluster.

The SPP is expected to be connected to a limited number of latency-critical peripherals that are shared among cores. Each core can access to the peripherals connected to SPP. The *ACE-Lite Subordinate* (ACELS) interface cannot access to the SPP.

**Note**

You can also connect the SPP to a shared interconnect but its latency advantages would be reduced.

You can connect the SPP to memory instead of peripherals although this is not optimal. It is because the SPP always applies Device non-Gathering, non-Reordering, with no Early Write Acknowledgement (Device-nGnRnE) memory type and attributes. This means that the Cortex®-R82 processor never speculatively accesses the SPP, reorders accesses to the SPP, forwards data from the SPP, or merges writes to the SPP. The SPP does not support exclusives and atomics.



If you connect an ideal memory system to the SPP, then the Cortex®-R82 processor sustains one store doubleword transaction (generated by basic STR instructions) each cycle to the SPP indefinitely resulting a throughput of 64-bit of store data per cycle.

To guarantee the minimum latency characteristics of the SPP, you need to reserve certain buffers in the Cortex®-R82 processor. For more information, see [8.14 Real-time considerations](#) on page 182.

You can connect the SPP to AXI4 subordinates, AXI5 subordinates, and AXI5-Lite subordinates. If you choose to connect the SPP to the AXI5-Lite subordinates, you must implement a bridge in your system to split the 2-beat bursts that SPP is capable of generating.

Unlike the *Main Manager* (MM) accesses, the SPP accesses are never shared or merged with accesses from a different context, even for Normal type memory. Therefore, it is always possible for the subordinate device to use the *Virtual Machine Identifier* (VMID) signaled on the SPP to restrict accesses to a particular context.

The SPP signals the VMID of the context that initiated a transaction on dedicated AXI User signals for all transactions to the SPP.

The SPP supports only INCR transactions and is capable of both single-beat and 2-beat bursts. For example, 128-bit load-store pair can be sent out as a 2-beat 64-bit INCR burst.

The SPP does not have exclusives or atomics support.

The IMP_SPPREGIONR_EL1 System register holds the information about:

- Whether the SPP is implemented or not.
- SPP region size which is fixed to 128MB if the SPP is implemented and 0 otherwise.
- The SPP base address.
- Whether the SPP is enabled or not.

SPP configuration parameter controls whether the SPP interface is implemented or not. If the system does not implement the SPP interface ($SPP = 0$), the Cortex®-R82 processor does not include the SPP region and ports. All related AXI signals, CFGSPPIMP, and CFGSPPBASEADDR pins are rendered out by the configuration script.

If the system implements the SPP interface ($SPP = 1$), the Cortex®-R82 processor includes the SPP region and ports. The SPP interface can be enabled or disabled by the CFGSPPIMP pin. If the SPP interface is implemented, the SPP region has a fixed size of 128MB.

The SPP base address is set via the configuration signal CFGSPPBASEADDR.

The SPP is disabled at reset. IMP_SPPREGIONR_EL1.ENABLEEL2 controls whether the SPP is enabled for access at EL2 and IMP_SPPREGIONR_EL1.ENABLEEL10 controls whether the SPP is enabled for access at EL0 and EL1.

For more information about the IMP_SPPREGIONR_EL1, see [A.2.2.43 IMP_SPPREGIONR_EL1, SPP Region Register](#) on page 510.

If the SPP is implemented and enabled, data accesses to an address in the SPP region are performed through the SPP. If the SPP is implemented and not enabled, data accesses to the SPP region generate a synchronous External abort. If the SPP is not implemented, then data accesses to the SPP region are performed through other parts of the memory system.

If the SPP is implemented, instruction accesses to an address in the SPP region cause a synchronous External abort. If the SPP is not implemented, instruction accesses are performed through other parts of the memory system.

The following table shows SPP attributes where NUM_CORES is the number of physical cores. See [1.3.1 Configuration parameters](#) on page 15 for permitted values.

Table 8-15: SPP attributes

Attribute	Value	Comments
Write issuing capability in SPLIT or HYBRID mode	NUM_CORES * 8	Each core can issue a maximum of eight write requests.
Write issuing capability in LOCK mode	NUM_CORES/2	Logical core.
Read issuing capability in SPLIT or HYBRID mode	NUM_CORES	Each core can issue one SPP read request.
Read issuing capability in LOCK mode	NUM_CORES/2	Logical core.
Combined issuing capability	NUM_CORES * 9	Each core can issue a maximum of nine requests with up to eight writes and one read.
Exclusive thread capability	0	No Exclusive access support.
Write ID capability	1 ID per core	All writes use the same ID per core. Note: Write ID capability is the same as the Read ID capability for the same core.
Write ID width	3 bits	Unused bits tied to zero if less than three bits required.
Read ID capability	1 ID per core	All reads use the same ID per core.
Read ID width	3 bits	Unused bits tied to zero if less than three bits required.

8.6.1 SPP features

The following table shows the *Shared Peripheral Port* (SPP) AXI properties the Cortex®-R82 processor supports or requires the cluster interconnect and system to support.

Table 8-16: SPP features

AXI property	Supported by the Cortex®-R82 processor SPP	Interconnect support required
Multi_Copy_Atomicity	Yes	Yes
Ordered_Write_Observation	Yes	Yes

AXI property	Supported by the Cortex®-R82 processor SPP	Interconnect support required
Atomic_Transactions	No	No
Poison	No	No
Check_Type	No	No
QoS_Accept	No	No
Trace_Signals	No	No
Loopback_Signals	No	No
Wakeup_Signals	Yes	Yes
Untranslated_Transactions	No	No
NSAccess_Identifiers	No	No

8.6.2 SPP memory attributes

The Cortex®-R82 processor does not implement AMBA® AXI5 ARCACHE and AWCACHE signals for the *Shared Peripheral Port* (SPP). The SPP uses the default values for these signals to indicate the memory attributes of transactions.

The following table shows the default encoding that is used for the ARCACHE and AWCACHE signals. Because the SPP is optimized for peripherals, all accesses propagate Device Non-bufferable attributes, regardless of the attributes that are returned by the memory management system translation. All other encodings are unused.

Table 8-17: ARCACHE and AWCACHE encodings

Encoding	Meaning
0b0000	Device Non-bufferable

8.6.3 SPP transfers

The SPP only issues INCR transactions of single-beat and 2-beat bursts and transfers of up to 128 bits per transfer.

The SPP read-channel can post one, up to 128-bit outstanding transaction per core when a single instruction requests 128 bits of data from a 128-bit aligned location. The SPP does not post two read transactions per core on the bus for two separate instructions.

The SPP write-channel generates up to eight write requests per core with up to 128-bit transactions on the bus from one or more instructions. All transactions use the same ID per core to ensure that ordering is maintained.

8.6.4 SPP AXI transfer restrictions

The SPP conforms to the AMBA® 5 AXI specification, but it does not generate all the AXI transaction types that the specification permits.

This section describes the types of AXI transactions that the SPP generates. If you are designing an AXI subordinate interface to work only with the Cortex®-R82 SPP, you can take advantage of these restrictions and the interface attributes to simplify the subordinate.

This section also contains tables that show some examples of the types of AXI burst that the Cortex®-R82 processor generates. However, because a particular type of transaction is not shown here does not mean that the Cortex®-R82 processor does not generate such a transaction.

You can connect the SPP to both AXI4 subordinates and AXI5-Lite subordinates in addition to AXI5 subordinates.

An AXI4 subordinate device that is connected to the SPP must be capable of handling every kind of transaction that the AXI4 specification permits, except where there is an explicit statement in this chapter that such a transaction is not generated. You must not infer any additional restrictions from the example tables given.

An AXI5-Lite subordinate device that is connected to the SPP must be capable of handling every kind of transaction that the AXI5-Lite specification permits, except where there is an explicit statement in this chapter that such a transaction is not generated. You must not infer any additional restrictions from the example tables given.



Note

If you connect the SPP to AXI5-Lite subordinates, then you must implement a bridge in your system to split the 2-beat bursts that SPP is capable of generating.

Restrictions on AXI peripheral transfers describes restrictions on the type of transfers that the SPP generates. You must not make any assumptions about the AXI handshaking signals, except that they conform to the AXI5 specification.

The SPP applies the following restrictions to the AXI transactions it generates:

- A burst never transfers more than 16 bytes.
- The burst length is never more than two transfers.
- No transaction ever crosses a 16-byte boundary in memory.
- All transactions are incrementing (INCR) bursts.
- Transactions to Device memory are always to addresses that are aligned to the transfer size (not to the transaction size).
- Does not support exclusive accesses and atomics.

8.6.4.1 SPP transactions

This section describes the Cortex®-R82 processor *Shared Peripheral Port* (SPP) transactions. The SPP Device memory and Normal memory transactions are the same.

The following table shows the values of ARADDRD, ARBURSTD, ARSIZED, and ARLEND for `LDRB` from bytes 0-7.

Table 8-18: LDRB transfers

Address[2:0]	ARADDRD	ARBURSTD	ARSIZED	ARLEND
0x0 (byte 0)	0x00	INCR	8-bit	1 data transfer
0x1 (byte 1)	0x01	INCR	8-bit	1 data transfer
0x2 (byte 2)	0x02	INCR	8-bit	1 data transfer
0x3 (byte 3)	0x03	INCR	8-bit	1 data transfer
0x4 (byte 4)	0x04	INCR	8-bit	1 data transfer
0x5 (byte 5)	0x05	INCR	8-bit	1 data transfer
0x6 (byte 6)	0x06	INCR	8-bit	1 data transfer
0x7 (byte 7)	0x07	INCR	8-bit	1 data transfer

The following table shows the values of ARADDRD, ARBURSTD, ARSIZED, and ARLEND for `LDRH` from halfwords 0-3.

Table 8-19: LDRH transfers

Address[2:0]	ARADDRD	ARBURSTD	ARSIZED	ARLEND
0x0 (halfword 0)	0x00	INCR	16-bit	1 data transfer
0x2 (halfword 1)	0x02	INCR	16-bit	1 data transfer
0x4 (halfword 2)	0x04	INCR	16-bit	1 data transfer
0x6 (halfword 3)	0x06	INCR	16-bit	1 data transfer

The following table shows the values of ARADDRD, ARBURSTD, ARSIZED, and ARLEND for an `LDR` transfer.

Table 8-20: LDR transfers

Address[2:0]	ARADDRD	AWBURSTD	ARSIZED	ARLEND
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer
0x0 (doubleword 0)	0x00	INCR	64-bit	1 data transfer

The following table shows the values of ARADDRD, ARBURSTD, ARSIZED, and ARLEND for an `LDP`.

All accesses using `LDP` instructions to memory occur as one or multiple 32-bit transactions or one or multiple 64-bit transactions.

Table 8-21: LDP transfers for two 32-bit registers

Address[2:0]	ARADDRD	ARBURSTD	ARIZED	ARLEND
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer
	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
	0x0C	INCR	32-bit	1 data transfer
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer
	0x08	INCR	32-bit	1 data transfer
	0x0C	INCR	32-bit	1 data transfer

Table 8-22: LDP transfers for two 64-bit registers

Address[2:0]	ARADDRD	ARBURSTD	ARIZED	ARLEND
0x0 (quadword)	0x00	INCR	64-bit	2 data transfers

The following table shows the values of AWADDRD, AWBURSTD, AWSIZED, AWLEND, and WSTRBD for an STRB from bytes 0-7 in memory.

Table 8-23: STRB transfers

Address[2:0]	AWADDRD	AWBURSTD	AWSIZED	AWLEND	WSTRBD
0x0 (byte 0)	0x00	INCR	8-bit	1 data transfer	0b00000001
0x1 (byte 1)	0x01	INCR	8-bit	1 data transfer	0b00000010
0x2 (byte 2)	0x02	INCR	8-bit	1 data transfer	00000b0100
0x3 (byte 3)	0x03	INCR	8-bit	1 data transfer	0b00001000
0x4 (byte 4)	0x04	INCR	8-bit	1 data transfer	0b00010000
0x5 (byte 5)	0x05	INCR	8-bit	1 data transfer	0b00100000
0x6 (byte 6)	0x06	INCR	8-bit	1 data transfer	0b01000000
0x7 (byte 7)	0x07	INCR	8-bit	1 data transfer	0b10000000

The following table shows the values of AWADDRD, AWBURSTD, AWSIZED, AWLEND, and WSTRBD for an STRB from halfwords 0-3 in memory.

Table 8-24: STRH transfers

Address[2:0]	AWADDRD	AWBURSTD	AWSIZED	AWLEND	WSTRBD
0x0 (halfword 0)	0x00	INCR	16-bit	1 data transfer	0b00000011
0x2 (halfword 1)	0x02	INCR	16-bit	1 data transfer	0b00001100
0x4 (halfword 2)	0x04	INCR	16-bit	1 data transfer	0b00110000
0x6 (halfword 3)	0x06	INCR	16-bit	1 data transfer	0b11000000

The following table shows the values of AWADDRD, AWBURSTD, AWSIZED, AWLEND, and WSTRBD for an STR to memory.

Table 8-25: STR transfers

Address[2:0]	AWADDRD	AWBURSTD	AWSIZED	AWLEND	WSTRBD
0x0 (word 0)	0x00	INCR	32-bit	1 data transfer	0b00001111
0x4 (word 1)	0x04	INCR	32-bit	1 data transfer	0b11110000
0x0 (doubleword 0)	0x00	INCR	64-bit	1 data transfer	0b11111111

The following table shows the values of AWADDRD, AWBURSTD, AWSIZED, AWLEND, and WSTRBD for an STR that writes two 64-bit registers, an STR <xt1>, <xt2>, to memory.

Table 8-26: STP transfers for two 64-bit registers

Address[2:0]	AWADDRD	AWBURSTD	AWSIZED	AWLEND	WSTRBD
0x0 (quadword)	0x00	INCR	64-bit	2 data transfer	0b11111111
					0b11111111

The following table shows the values of AWADDRD, AWBURSTD, AWSIZED, AWLEND, and WSTRBD for an STP that writes two 32-bit registers, an STP <Rt1>, <Rt2>, to memory.

Table 8-27: STP transfers for two 32-bit registers

Address[2:0]	AWADDRD	AWBURSTD	AWSIZED	AWLEND	WSTRBD
0x0	0x00	INCR	64-bit	1 data transfer	0b11111111
0x4	0x04	INCR	32-bit	2 data transfer	0b11110000
0x6					0b00001111

For the *Shared Peripheral Port* (SPP) transactions:

- A load of a halfword from memory addresses 0x1 or 0x3 generates an alignment fault.
- A load of a word from memory addresses 0x1, 0x2, or 0x3 generates an alignment fault.
- A load of a doubleword from memory addresses 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, or 0x7 generates an alignment fault.
- A 32-bit load-pair from memory addresses 0x1, 0x2, 0x3, 0x5, 0x6, or 0x7 generates an alignment fault.
- A 64-bit load-pair from memory addresses 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, or 0x7 generates an alignment fault.
- A store of a halfword from memory addresses 0x1, or 0x3 generates an alignment fault.
- A store of a word to memory addresses 0x1, 0x2, or 0x3 generates an alignment fault.
- A store of a doubleword from memory addresses 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, or 0x7 generates an alignment fault.
- A 32-bit store-pair to memory address 0x1, 0x2, 0x3, 0x5, 0x6, or 0x7 generates an alignment fault.



Note

- A 64-bit store-pair to memory address 0x1, 0x2, 0x3, 0x4 0x5, 0x6, or 0x7 generates an alignment fault.
-

8.7 MM interface

The *Main Manager* (MM) interface is the default memory interface of the Cortex®-R82 processor. The MM interface is optimized for highest average performance for contexts where determinism is less critical. However, a real-time context is also able to access the MM port, although such an access might not be desirable depending on the system design.

The MM interface is used for all the memory space that is not associated with another interface.

The MM interface is used for accesses to high-latency memory such as DDR and non-critical peripherals that are shared between cores within the cluster and other agents in the system.

You are expected to connect the MM port to an interconnect.

You can configure the MM port as:

- A 256-bit AMBA® CHI.E requester that can be connected to a CHI.E subordinate.
- A 256-bit AMBA® 5 AXI manager interface that can be connected to either an AMBA® 4 AXI or an AMBA® 5 AXI subordinate interface.

The MM port is accessible by external agents through the *Main Accelerator Coherency Port* (MACP) subordinate interface.

Accesses to the MM port can be cached in both the L1 (data caches and instruction caches) and L2 cache as determined by memory type and attributes. The Cortex®-R82 processor supports only Write-Back caching of MM addresses. MM addresses that are marked as Write-Through cacheable are treated as Non-cacheable.

The Cortex®-R82 processor includes hardware coherency logic for addresses in the MM address ranges so that software does not need to maintain cache coherency for shared data. The Cortex®-R82 processor coherency hardware automatically updates the contents of L1 data and L2 caches within the cluster to ensure all cores within the cluster have the same coherent view of memory (full coherency).

The coherency hardware also provides coherency with non-cached external agents accessing the MM port through the MACP subordinate interface. The coherency hardware automatically updates the contents of caches within the cluster but is not able to update the content of managers connected to the MACP (I/O coherency).

When the MM port is configured as a CHI requester, the coherency hardware also provides coherency with external coherent agents connected through a coherent interconnect. The coherency hardware automatically processes CHI coherency requests and update the contents of caches within the cluster. The coherency hardware also generates CHI coherency requests on the MM port for the coherent system to process.

When configured as a CHI requester, the Cortex®-R82 processor MM interface accepts and processes stashing requests to the L2 cache. This enables an external coherent requester to request the Cortex®-R82 processor to fetch a cache line on the MM port and allocate it in the L2 cache.

The Cortex®-R82 processor MM interface supports atomic instructions and performs all the atomics either in the L1 data cache or in the interconnect outside the Cortex®-R82 processor if the interconnect supports the atomics.

The Cortex®-R82 processor does not include logic to perform atomic operations at the cluster level. Whether the atomic instructions are handled within the core or atomic transactions are generated on the MM port depends on:

- The memory shareability attributes.

This allows you to design systems where:

- Atomic transactions are generated on the MM if BROADCASTATOMIC is set to 0
- Atomic transactions are generated on the MM and the Cortex®-R82 processor cluster is just one of the number of agents accessing a shared location.

8.7.1 CHI requester interface

You can configure the Cortex®-R82 processor to use the AMBA® 5 CHI protocol for the *Main Manager* (MM) interface.

The Cortex®-R82 processor supports CHI Issue E.

8.7.1.1 CHI features

AMBA® defines a set of interface properties for the CHI interconnect. The following table shows which of these properties the Cortex®-R82 processor supports, or requires the interconnect and system to support.

Table 8-28: CHI interconnect properties for the Cortex®-R82 processor

CHI property	Supported by the Cortex®-R82 processor	Interconnect support required
Atomic_Transactions	Yes if BROADCASTATOMICM is HIGH.	Yes if BROADCASTATOMICM is HIGH.
Cache_Stash_Transactions	Yes	Yes
Direct_Memory_Transfer	Yes	Optional. The Cortex®-R82 processor supports this feature if it is implemented by the interconnect.
Direct_Cache_Transfer	Yes	Optional. The Cortex®-R82 processor supports this feature if it is implemented by the interconnect.
Data_Poison	Yes, cache protection (CORE_RAM_PROTECTION) is always present.	Yes, cache protection (CORE_RAM_PROTECTION) is always present.

CHI property	Supported by the Cortex®-R82 processor	Interconnect support required
Data_Check	No	No
Check_Type	No	No
CCF_Wrap_Order	Yes	No
Req_Addr_Width	48	Not applicable
NodeID_Width	11	Not applicable
Data_Width	256 bits	Not applicable
Barrier_Transactions	Yes	Yes
Data return from SC state.	Yes	Configurable via the CLUSTERACTLR using the fields L2FLUSHEVC and L2CDEVC
I/O de-allocation transactions (ROMI and ROCI).	No	No. The Cortex®-R82 processor does not use these transaction types.
ReadNotSharedDirty transactions	Yes	Yes
CleanSharedPersist transactions	Yes if BROADCASTPERSISTM is HIGH.	Yes if BROADCASTPERSISTM is HIGH.
Receiving of Forwarding snoops	Yes	Yes

For more information on these features, see [AMBA® 5 CHI Architecture Specification](#).

8.7.1.2 CHI configurations

You can change the coherency configurations to suit your system configuration using the BROADCASTCACHEMAINTM and BROADCASTOUTERM input signals.

CHI configurations

The following table shows the permitted combinations of these signals and the supported configurations in the Cortex®-R82 processor with a CHI bus.

Table 8-29: Supported CHI configurations

Signal	Feature			
	CHI non-coherent		CHI coherent	
	With no cache or invisible system cache	With visible system cache	With invisible system cache	With visible system cache
BROADCASTCACHEMAINTM	0	1	0	1
BROADCASTOUTERM	0	0	1	1
BROADCASTPERSISTM	0	1	0	1 Note: BROADCASTOUTERM should be set or BROADCASTCACHEMAINTM should be set. Or BROADCASTPERSISTM and a CleanSharedPersist.
BROADCASTATOMICM	0	1	0	1

Signal	Feature			
	CHI non-coherent		CHI coherent	
	With no cache or invisible system cache	With visible system cache	With invisible system cache	With visible system cache
BROADCASTICINVALM	0	1	0	1 Note: Blocks IC DVMs from going external.
BROADCASTTLBIINNERM	0	1	0	1
BROADCASTTLBIOUTERM	0	1	0	1



Note

- A visible system cache requires cache maintenance transactions to ensure that a write is visible to all observers.
- An invisible system cache is one that does not require cache maintenance transactions to ensure that a write is visible to all observers. This is true even if those observers use different memory attributes.

The following table shows the key features in each of the supported CHI configurations.

Table 8-30: Supported features in the CHI configurations

Features	Configuration		
	CHI non-coherent		CHI coherent
	With no cache or invisible system cache	With visible system cache	
Cache maintenance requests on TXREQ channel	No	Yes	Yes
Snoops on RXSNP channel	No	No	Yes
Coherent requests on TXREQ channel	No	No	Yes
DVM requests on TXREQ channel	No	No	Yes

The input signals BROADCASTTLBIINNER and BROADCASTTLBIOUTER control the broadcasting of *TLB Invalidate* (TLBI) DVM messages to the external interconnect. The following table shows how the broadcast of the TLBI messages are controlled for the Inner and Outer Shareable domains depending on the configuration of BROADCASTTLBIINNER and BROADCASTTLBIOUTER.

Table 8-31: Control of Inner and Outer Shareable TLBI messages to the external interconnect

BROADCASTTLBIINNER	BROADCASTTLBIOUTER	Description
LOW	LOW	No TLBI transactions are broadcast outside the cluster
LOW	HIGH	Outer Shareable TLBI transactions, TLBI {OS}, generate TLBI transactions that are broadcast from the cluster. No other TLBI instructions generate TLB transactions that are broadcast from the cluster.
HIGH	LOW	Invalid configuration

BROADCASTTLBIINNER	BROADCASTTLBIOUTER	Description
HIGH	HIGH	Inner shareable TLBI instructions, TLBI {IS}, and Outer shareable TLBI instructions, TLBI {OS}, generate TLBI transactions that are broadcast from the cluster.

Cluster Power Mode Blocking

It should be noted that tying the CHI RXSACTIVE signal high then it may not be possible for the cluster to enter a lower power mode, for example, ON to OFF or ON to MEM_RET. A stream of external snoop requests can also prevent a power transition to a lower power mode from completing until the stream of external snoops and outstanding snoops have completed.

8.7.1.3 CHI requester interface attributes

The following table lists the possible values for the read and write issuing capabilities.

Table 8-32: Attributes of the CHI requester interface

Attribute	Value	Comments
Write issuing capability	Configuration dependent	The maximum number of writes is: <ul style="list-style-type: none"> 26, if L2_SLICES = 1. 52, if L2_SLICES = 2
Read issuing capability	Configuration dependent	The maximum number of reads is: <ul style="list-style-type: none"> $(\text{NUM_CPUS} * 8) + (16 * \text{L2_SLICES})$ if L2 cache is not implemented $(\text{NUM_CPUS} * 10) + (16 * \text{L2_SLICES})$ if L2 cache is implemented <p>Note: Two-part Distributed Virtual Memory (DVM) messages use the same ID for both parts, and therefore can have two outstanding transactions on the same ID.</p> <p>Note: In some cases, the L2 cache may be able to exceed this read acceptance capability. However this is rare and should not be used when sizing your interconnect.</p>
Exclusive hardware access thread capability	Number of hardware threads.	Each hardware thread can have one exclusive access sequence in progress.
Transaction ID width	12 bits	There is no fixed mapping between CHI transaction IDs and cores. Transaction IDs can be used for either reads or writes. <p>Note: The source of the transaction is encoded in the LPID field, see Table 8-34: CHI LPID assignment on page 147.</p>

Attribute	Value	Comments
Transaction ID capability	Configuration dependent	<p>The ID capability is:</p> <ul style="list-style-type: none"> (NUM_CPUS * 8) + (16 * L2_SLICES) if L2 cache is not implemented (NUM_CPUS * 10) + (16 * L2_SLICES) if L2 cache is implemented <p>Note: Unlike in an AMBA® ACE5 configuration, there is never any ID reuse in CHI implementations, regardless of the memory type.</p>
NodeID widths	11 bits	-
TXREQFLITM.RSVDC	0 bits	-
TXDATFLITM.RSVDC	0 bits	-
TXDATFLITM.DataCheck	0 bits	-

**Note**

The issuing capability described here is the maximum for the whole cluster, and can be used to size interconnect capabilities if you want to achieve the maximum performance available. However a single core may not reach this maximum on its own. It may need multiple cores generating heavy memory traffic simultaneously to reach maximum performance. The capabilities vary by memory type, with typically a significantly lower limit for Device or Non-cacheable transactions than for Cacheable transactions.

8.7.1.4 CHI channel properties

The following table shows the snoop capabilities and other CHI channel properties for the Cortex®-R82 processor.

Table 8-33: CHI channel properties

Property	Value	Comment
Snoop acceptance capability	Configuration dependent	The L2 coherency logic can accept and process a maximum of 9 snoop requests from the system.
DVM acceptance capability	4	<p>The L2 coherency logic can accept and process a maximum of four <i>Distributed Virtual Memory</i> (DVM) transactions from the system. Each of these four transactions can be a two part DVM message.</p> <p>The interconnect must be configured to never send more than four DVM messages to the cluster, otherwise the system might deadlock.</p>
Snoop latency	Hit	<p>When there is a hit in L2 cache, the best case for response and data is 10 SCLK cycles. When there is a miss in the L2 cache but a hit in an L1 cache in a core, then the latency varies. This latency variation depends on the type and configuration of that core.</p> <p>Latencies can be higher if hazards occur or if there are not enough buffers to absorb requests.</p>
	Miss	Best case for latency is six SCLK cycles when the snoop filter and L2 cache tags indicate the miss.
	DVM	The cluster takes a minimum of six SCLK cycles to provide a response to DVM packets.

Property	Value	Comment
Snoop filter	Supported	<p>The cluster provides support for an external snoop filter in an interconnect. It indicates when clean lines are evicted from the cluster by sending Evict transactions on the CHI write channel.</p> <p>However there are some cases that can prevent an Evict transaction from being sent. Therefore you must ensure that you build any external snoop filter to handle a capacity overflow. When exceeding capacity, the snoop filter should send a back-invalidation to the cluster.</p> <p>Examples of case where evicts are not produced include:</p> <ul style="list-style-type: none"> • Linefills that take External aborts. • Store exclusives that fail. • Mis-matched aliases.
Supported transactions	-	<p>All transactions that are described by the CHI protocol:</p> <ul style="list-style-type: none"> • Are accepted on the CHI requester interface from the system. • Can be produced on the CHI requester interface except: <ul style="list-style-type: none"> ◦ PCRDReturn ◦ WriteNoSNPDef ◦ WriteUniqueFullClearShared ◦ WriteUniqueFullCleanSharedPersistStep ◦ WriteNoSnoopPTLCleanShared ◦ WriteNoSnoopPTLCleanValid ◦ WriteNoSnoopPTLCleanSharedPersistSEP ◦ WriteUniquePTLCleanShared ◦ WriteUniquePTLCleanSharedPersitsSEP ◦ WriteBackCleanSharedPersistSEP ◦ WriteBackCleanShared ◦ WriteNoSNPPTLCleanInvPOPA ◦ WriteNoSNPFullCleanInvPOPA ◦ WriteUniqueEPTLStash ◦ StashOnCESPShared ◦ StashOnCESPUnique ◦ PrefetchtGT

8.7.1.5 CHI transactions

CHI transactions are sent to a specific node in the interconnect that is based on the following criteria:

- Type of access.
- Address of the access.
- Settings of the System Address Map.

Addresses that map to an HN-F node can be marked as cacheable memory in the translation tables, and can take part in the cache coherency protocol. Addresses that map to an HN-I or MN must be marked as device or Non-cacheable memory.

CHI TXREQ transactions include the *Logical Processor ID* (LPID) field. This field uniquely identifies the logical core that generated the request transaction. The following table shows CHI LPID assignment.

Table 8-34: CHI LPID assignment

LPID	Description
0x00-0x07	CPUID
0x08	ACP request.

Table 8-35: CHI transaction types

Transaction	Operation
ReadNoSnP	Non-cacheable loads or instruction fetches. Linefills of Non-shareable cache lines into L1 or L2 caches.
ReadOnce	Cacheable loads that are not allocating into the cache.
ReadNotSharedDirty	Cache data linefills started by a load instruction, or cache linefills started by an instruction fetch.
ReadUnique	Cache data linefills started by a store instruction.
CleanUnique	Store instructions that hit in the cache but the line is not in a unique coherence state.
MakeUnique	Store instructions of a full cache line of data, that miss in the caches.
MakeReadUnique	Read request to a Snooperable address region requesting a unique copy of a cache line.
CleanShared	Cache maintenance instructions.
CleanSharedPersistSep	Cache maintenance instructions. This is only generated by the <i>Data Cache Clean to the Point of Persistence</i> (DC_CVAP) cache maintenance instruction when the BROADCASTPERSISTM input signal is HIGH.
CleanInvalid	Cache maintenance instructions.
DVMOp	TLB and instruction cache maintenance instructions.
EOBarrier	Not used.
ECBarrier	Not used.
PrefetchTgt	Hardware prefetch hint to the memory controller.
StashOnceShared	Cache prefetch when there is no L2 cache present.
StashOnceUnique	Cache prefetch when there is no L2 cache present.
WriteNoSnPPtl	Non-cacheable store instructions.
WriteNoSnPFull	Non-cacheable store instructions, or evictions of Non-shareable cache lines.
WRITENOSNPCLEANSHARED	Combined WRITENOSNP with CleanShared CMO.
WRITENOSNPCLEANINVALID	Combined WRITENOSNP with CleanInvalid CMO.
WriteUniqueFull	Cacheable writes of a full cache line, that are not allocating into L1 or L2 caches, for example streaming writes.
WriteUniquePtl	Not used.
WriteBackFull	Evictions of dirty lines from the L1 or L2 caches.
WRITEBACKFULLCLEANINVALID	Combined WRITEBACKFULL with CLEANINVALID CMO
WriteBackPtl	Not used.

Transaction	Operation
WriteCleanFull	Evictions of dirty lines from the L2 cache, when the line is still present in an L1 cache. Some cache maintenance instructions.
WRITECLEANFULLCLEANSHARED	Combined WRITECLEANFULL with CLEANSHARED CMO.
WriteCleanPtl	Not used.
WriteEvictFull	Evictions of unique clean lines, when configured in the IMP_CLUSTERACTLR_EL1.
WRITEEVICTOREVICT	WriteBack of Clean data to the next-level cache. This request type is merging of WriteEvictFull and Evict into one request.
Evict	Evictions of clean lines, when configured in the IMP_CLUSTERACTLR_EL1.
AtomicStore	Atomic instruction.
AtomicLoad	Atomic instruction.
AtomicSwap	Atomic instruction.
AtomicCompare	Atomic instruction.
WriteUniqueFullStash	Not used.
WriteUniquePtlStash	Not used.
ReadOnceCleanInvalid	Not used.
ReadOnceMakeInvalid	Not used.

External memory accesses generate the following transactions in an implementation configured with a CHI requester interface.

Table 8-36: CHI transaction usage

Attributes		CHI transaction				
Memory type	Shareability	SnpAttr	Load	Store	Load exclusive	Store exclusive
Device.	Outer Shareable	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp and Excl set to HIGH.	WriteNoSnp and Excl set to HIGH.
Normal, Inner Non-cacheable, Outer Non-cacheable.	Non-shareable	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp and Excl set to HIGH.	WriteNoSnp and Excl set to HIGH.
	Inner Shareable					
	Outer Shareable					
Normal, Inner Non-cacheable, Outer Write-Back or Write-Through, or Normal, Inner Write-Through, Outer Write-Back, Write-Through or Non-cacheable, or Normal Inner Write-Back Outer Non-cacheable or Write-Through.	Non-shareable	Non-snoopable	ReadNoSnp	WriteNoSnp	ReadNoSnp and Excl set to HIGH.	WriteNoSnp and Excl set to HIGH.
	Inner Shareable					
	Outer Shareable					
Normal, Inner Write-Back, Outer Write-Back.	Non-shareable	Non-snoopable	ReadNoSnp	WriteNoSnp when the line is evicted or if not allocating into the cache.	ReadNoSnp	WriteNoSnp when the line is evicted.

Attributes		CHI transaction				
Memory type	Shareability	SnpAttr	Load	Store	Load exclusive	Store exclusive
	Inner Shareable	Snoopable	ReadNotSharedDirty	ReadUnique, CleanUnique, or MakeUnique if allocating into the cache, then a WriteBackFull when the line is evicted. WriteUniqueFull if not allocating into the cache.	ReadNotSharedDirty, with Excl set to HIGH.	CleanUnique with Excl set to HIGH if required, then a WriteBackFull when the line is evicted.
	Outer Shareable	Snoopable				

8.7.1.6 Use of DataSource

Some CHI responses from the interconnect include a DataSource field indicating where the data was supplied from.

When making use of the DataSource field, Arm recommends providing this information as accurately as possible using the encodings recommended in the table *Suggested DataSource value encodings* provided in the [AMBA® 5 CHI Architecture Specification](#).

The value of this field is used to calculate some *Performance Monitoring Unit* (PMU) events. It can also be used by some processors to tune the performance of their data prefetchers.

8.7.1.7 Support for memory types

The Cortex®-R82 processor simplifies the coherency logic by downgrading some memory types.

Normal memory that is marked as both Inner Write-Back Cacheable and Outer Write-Back Cacheable is cached in the core data caches and the L2 cache.

All other Normal memory types are treated as Non-cacheable and are sent on the MM bus interface as Normal Non-cacheable.

8.7.2 AXI manager interface

The AXI *Main Manager* (MM) port is a 256-bit AMBA® 5 AXI manager that can be connected to either an AMBA® 4 AXI or an AMBA® 5 AXI subordinate interface.

When configured as an AXI manager, the MM port implements the following signals that are specified in AXI5 but not in AXI4:

- Signal integrity protection signals. These signals are intended to provide point-to-point protection between the Cortex®-R82 processor cluster and the interconnect and can be transmitted through Data Check signals.
- Atomic operation signal, BROADCASTATOMICM. This signal is used to enable or disable broadcasting of atomic instructions on the MM interface.
- This can be achieved by tying CFGMMPOISON to 0 during the integration of the Cortex®-R82 processor into your system. From the Cortex®-R82 processor register definition, when such field is set to 0 Main Manager port does not support poison.

The MM port implements the atomic operation signal, BROADCASTATOMICM, that is specified in AXI5 but not in AXI4. This signal is used to enable or disable broadcasting of atomic instructions on the MM interface.

The Cortex®-R82 processor includes coherency hardware that automatically manages the contents of caches within the cluster to ensure all cores have a coherent view of AXI MM addresses.

When configured as an AXI manager, the MM port does not support:

- Barriers on AR and AW channels.
- Cache maintenance requests on AR channel.
- Snoop capabilities.

8.7.2.1 MM AXI features

AMBA® defines a set of interface properties for the AXI interconnect. The following table shows which of these properties the Cortex®-R82 processor *Main Manager* (MM) interface supports or requires the cluster interconnect and system to support.

Table 8-37: AXI interconnect properties for the Cortex®-R82 processor

AXI property	Supported by the Cortex®-R82 processor MM	Interconnect support required
Multi_Copy_Atomicity	Yes	Yes
Ordered_Write_Observation	Yes	Yes
Atomic_Transactions	Yes	Yes
Poison	Yes	Yes
Check_Type	No if bus protection is not included (BUS_PROTECTION 0). Odd_Parity_Byte_All if bus protection is included.	Yes
QoS_Accept	No	No
Trace_Signals	No	No
Loopback_Signals	No	No
Wakeup_Signals	Yes	Yes
Untranslated_Transactions	No	No

AXI property	Supported by the Cortex®-R82 processor MM	Interconnect support required
NSAccess_Identifiers	No	No

8.7.2.2 MM AXI attributes

The following table lists the read and write issuing capabilities for the *Main Manager* (MM) AXI interface.

Table 8-38: AXI manager interface attributes

Attribute	Value	Comments
Write issuing capability	Configuration dependent	The maximum number of writes is: <ul style="list-style-type: none"> 13, if the L2 is configured with a single slice 26, if configured with two slices
Read issuing capability	<ul style="list-style-type: none"> (NUM_CORES * 8) + 4, if L2 cache is not implemented (NUM_CORES * 10) + 4, if L2 cache is implemented 	<p>The maximum number of reads is:</p> <ul style="list-style-type: none"> 68 if L2 cache is not implemented 84 if L2 cache is implemented <p>Note: Two-part <i>Distributed Virtual Memory</i> (DVM) messages use the same ID for both parts, and therefore can have two outstanding transactions on the same ID.</p> <p>Note: In some cases, the L2 cache may be able to exceed this read acceptance capability. However this is rare and should not be used when sizing your interconnect.</p>
Combined issuing capability	Configuration dependent (NUM_CORES and L2 cache implemented or not)	The maximum combined issuing capability is 116 (Eight cores and L2 cache implemented).
Exclusive thread capability	Number of hardware threads, maximum eight threads (one per core)	Each hardware thread can have one exclusive access sequence in progress.
Write ID capability	Configuration dependent	<p>The maximum write ID capability is:</p> <ul style="list-style-type: none"> 57, if less than four cores are present. 89, if four or more cores are present. <p>Only Device memory types with nGnRnE or nGnRE can have more than one outstanding transaction with the same AXI ID. All other memory types use a unique AXI ID for every outstanding transaction.</p>
Write ID width	13	The ID encodes the source of the memory transaction. See the Encodings for AWIDM[12:0] table.

Attribute	Value	Comments
Read ID capability	Configuration dependent	<p>The maximum read ID capability is:</p> <ul style="list-style-type: none"> 185, if less than four cores are present. 345, if four or more cores are present. <p>Only Device memory types with nGnRnE or nGnRE can have more than one outstanding transaction with the same AXI ID. All other memory types use a unique AXI ID for every outstanding transaction.</p> <p>Two part DVMs use the same ID for both parts, and therefore can have two outstanding transactions on the same ID.</p>
Read ID width	13	The ID encodes the source of the memory transaction. See the Encodings for ARIDM[12:0] table.



The issuing capability described here is the maximum possible for the whole cluster. This can be used to size interconnect capabilities if you want to achieve the maximum performance available. However, this maximum may not be reached depending on the Cortex®-R82 processor configuration and characteristics of your system.

The following table shows the encodings for AWIDM[12:0], ARIDM[12:0].

Table 8-39: Encodings for AWIDM[12:0]

Attribute	Value	Issuing capability per ID	Description	Comments
Write ID	0b00nnn00000000	1	System domain store exclusives, excluding Device non-Reordering	nnn = Core ID
	0b11ssss0000000	15	Non-Reordering Device writes	ssss = Subordinate ID
	0b10nnncr0ammm	1	All other writes	<p>nnn = Core ID</p> <p>c = L1 colour</p> <p>r = Slice ID</p> <p>mmmm = L2DB ID</p> <p>a = ACP, HWFLUSH or padding</p>

The following table shows the Encodings for ARIDM[12:0].

Table 8-40: Encodings for ARIDM[12:0]

Attribute	Value	Issuing capability per ID	Description	Comments
Read ID	0b00nnn00000000	1	Load exclusives, excluding Device Non-Reordering	nnn = Core ID
	0b11ssss0000000	17	Non-Reordering Device reads	sss = Subordinate ID

Attribute	Value	Issuing capability per ID	Description	Comments
	0b10nnncr0ammm	1	READONCE and <i>Main Accelerator Coherency Port (MACP)</i> reads, Atomic read data	nnn = Core ID/ ACP ID c = L1 color or ACP poison r = Slice ID mmmm = L2DB ID a = ACP, HWFLUSH or padding
	0b01nnnrpppppp	1	All other reads	nnn = Core ID r = Slice ID ppppppp = Internal request ID

nnn is the core number 0b000-0b111 in binary.



These ID and transaction details are provided for information only. Arm strongly recommends that all interconnects and peripherals are designed to support any type and number of transactions on any ID, to ensure compatibility with future products.

For more information about the AXI signals described in this manual, see the [AMBA® AXI Protocol Specification](#).

8.7.2.3 MM AXI transactions

The Cortex®-R82 processor does not generate any FIXED bursts and a burst does not cross a cache line boundary.

The cache linefill fetch length is always 64 bytes.

The Cortex®-R82 processor generates only a subset of all possible AXI transactions on the manager interface.

For Write-Back Cacheable transfers, the supported transfers are:

- WRAP 2 256-bit for read transfers (linefills).
- INCR 2 256-bit for write transfers (evictions).
- INCR 2 256-bit for read transfers (linefills).
- INCR 1 256-bit for write transfers.

For Normal Non-cacheable or Device transactions:

- INCR 2 256-bit read transfers.

- INCR 2 256-bit write transfers.
- WRAP 2 256-bit read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, 128-bit, and 256-bit read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, 128-bit, and 256-bit write transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit exclusive read transfers.
- INCR 1 8-bit, 16-bit, 32-bit, 64-bit, and 128-bit exclusive write transfers.

The following points apply to AXI transactions:

- WRAP bursts are only 256-bit size.
- INCR burst, more than one transfer, are only 256-bit size.
- No transaction is marked as FIXED.
- Write transfers with none, some, or all byte strobes LOW can occur.

8.7.2.4 Support for memory types

The Cortex®-R82 processor simplifies the coherency logic by downgrading some memory types.

Normal memory that is marked as both Inner Write-Back Cacheable and Outer Write-Back Cacheable is cached in the core data caches and the L2 cache.

All other Normal memory types are treated as Non-cacheable and are sent on the MM port as Normal Non-cacheable.

8.7.2.5 Write response

The AXI manager always accepts write responses without delay by holding BREADYM HIGH.

8.7.2.6 AXI4 compatibility mode

The Cortex®-R82 processor implements an AXI4 compatibility mode that enables you to use the Cortex®-R82 processor in a standalone environment where the AMBA® AXI5 interface is not required.

To enable this mode, you must ensure that the BROADCASTATOMICM signal is LOW.

8.7.2.7 MM AXI privilege information

The *Main Manager* (MM) interface provides information to indicate whether the request is Privileged or Unprivileged on the ARPROTM[0] and AWPROTM[0] signals.

Where a request cannot be merged (Loads and Stores to Device memory, or Non-cacheable Store Exclusives) the ARPROTM[0] and AWPROTM[0] reflect the privilege state of the requestor. Where requests might have been merged, including all Non-cacheable and cacheable loads and stores, the ARPROTM[0] and AWPROTM[0] indicate that the request is privileged.

The MM interface provides information about the Secure or Non-secure access on the ARPROTM[1] and AWPROTM[1] signals. The values of 0 indicates the access is Secure and the value of 1 indicates the access is Non-secure.

The MM interface indicates whether a request is for a data or instruction fetch via the ARPROTM[2] and AWPROTM[2] signals.

8.8 LLRAM manager interface

The *Low-latency RAM* (LLRAM) port provides deterministic, low-latency access to external memory shared between cores within the cluster. The Cortex®-R82 processor has an optional LLRAM manager interface that is shared among the cores within the cluster.

The LLRAM manager interface is a 256-bit AMBA® 5 AXI manager that can be connected to either an AMBA® 4 AXI or an AMBA® 5 AXI subordinate interface.

The LLRAM port does not implement any signals that are not specified in AXI5.

The LLRAM port implements the following signals that are specified in AXI5 but not in AXI4:

- Atomic operation signal, AWATOPPL. This signal is used to generate atomic transactions when CFGLLRAMSHARED = 1 and BROADCASTATOMICL = 1.
- Poison signaling

The LLRAM port is expected to have better real-time characteristics compared to the *Main Manager* (MM) port. Because the LLRAM port is shared among the cluster cores, it is expected to have worse real-time characteristics compared to *Tightly Coupled Memories* (TCMs) of individual cores.

To guarantee the deterministic, low-latency characteristics of the LLRAM port, you need to reserve certain buffers in the Cortex®-R82 processor. For more information, see [8.14 Real-time considerations](#) on page 182.

The LLRAM interface supports accesses to peripherals and also connecting to an interconnect. However, it is not optimized for such accesses. The LLRAM port is mainly used for:

- Code or read/write data shared among cores within the cluster.
- Data sharing between cores within the cluster (producer-consumer).

- Data sharing between cores within the cluster and external agents such as DMA through the *LLRAM Accelerator Coherency Port* (LLRAM ACP, implemented by the *ACE-Lite Subordinate* (ACELS) port).

Accesses to the LLRAM port can only be cached in the L1 data caches and L1 instruction caches as determined by memory type and attributes. The Cortex®-R82 processor L1 data caches support only Write-Through caching for LLRAM addresses. LLRAM addresses that are marked as Write-Back cacheable are treated as Write-Through.

Accesses to the LLRAM port cannot be cached in the L2 cache. The LLRAM port supports downstream System caches and propagates Cacheable attributes downstream.

LLRAM configuration parameter controls whether the LLRAM interface is implemented or not. If the system does not implement the LLRAM interface (LLRAM = 0), the Cortex®-R82 processor does not include the LLRAM region, port, and coherency logic. All related AXI signals, CFGLLRAMIMP, CFGLLRAMEN, and CFGLLRAMBASEADDR pins are rendered out by the configuration script.

If the system implements the LLRAM interface (LLRAM = 1), the Cortex®-R82 processor includes the LLRAM region, port, and coherency logic. The LLRAM interface can be enabled or disabled by the CFGLLRAMIMP pin and its behavior can be further defined by the CFGLLRAMEN pin. If the LLRAM interface is implemented, the LLRAM region has a fixed size of 256MB.

**Note**

If the LLRAM is not implemented (LLRAM = 0) or it is implemented but disabled (LLRAM = 1 and CFGLLRAMIMP tied LOW), associated *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs are not needed to be integrated. See *Arm® Cortex®-R82 Processor Configuration and Integration Manual* for more information.

The LLRAM interface has a size-aligned base address that is set by the configuration signal CFGLLRAMBASEADDR.

The Cortex®-R82 processor includes hardware coherency logic for addresses in the LLRAM address range so that software does not need to maintain cache coherency for the shared data. The Cortex®-R82 processor coherency hardware automatically updates the contents of L1 data caches within the cluster to ensure all cores within the cluster have the same coherent view of memory (full coherency).

The coherency hardware also provides coherency with non-cached external agents accessing the LLRAM port through the LLRAM ACP subordinate interface implemented by the ACELS port. The coherency hardware automatically updates the contents of L1 data caches within the cluster but is not able to update the content of managers connected to the LLRAM ACP (I/O coherency).

The LLRAM port is accessible by external agents through the LLRAM ACP subordinate interface. Although it is possible to use an interconnect, this removes the need to add an interconnect between the LLRAM port and the SRAM controller in systems that connect the LLRAM port directly to an SRAM controller. The LLRAM ACP shares the same physical port, ACELS, as the TCM subordinate port.

If the LLRAM is implemented and not enabled, data accesses to the LLRAM region generate a synchronous External abort.

The Cortex®-R82 processor LLRAM manager interface supports atomic instructions. The cluster includes logic to perform atomic operations. Whether atomic transactions are generated on the LLRAM port depends on the memory Shareability attributes and the value of CFGLLRAMSHARED and BROADCASTATOMICL signals. This enables you to design systems where the access to memory sub-system is either exclusive to Cortex®-R82 processor LLRAM port or shared with other manager ports in the system.

In systems where the Cortex®-R82 processor LLRAM port is the only manager port accessing its memory sub-system components (CFGLLRAMSHARED tied LOW):

- The LLRAM addresses are marked as Non-shareable, Inner Shareable, or Outer Shareable and can be cached in the L1 caches.
- All exclusives and atomic transactions are handled within the cluster.
- No exclusives or atomic transactions are generated on the LLRAM port.

In systems where the Cortex®-R82 processor LLRAM port is just one of the manager ports into an interconnect and access to the memory sub-system components is shared (CFGLLRAMSHARED tied HIGH):

- The LLRAM addresses that are marked as Outer Shareable cannot be cached in the L1 caches.
- The LLRAM addresses that are marked as Non-shareable or Inner Shareable can be cached in the L1 caches.
- Exclusives transactions can be generated on the LLRAM port.
- Atomic transactions can be generated on the LLRAM port if the BROADCASTATOMICL is HIGH.



Although the LLRAM region can be accessed by EL1 *Virtual Memory System Architecture* (VMSA) contexts, page tables are not allowed to be stored in the LLRAM, and must be stored in memory connected to the MM port instead. If the operating system you are running cannot guarantee that page tables are not placed in the LLRAM region, then you must ensure that the LLRAM region is either disabled or that the operating system does not use the LLRAM region at all. For example, in Linux you can use the *Device Tree* (DT) to specify that the address space occupied by the LLRAM region is reserved memory.

8.8.1 LLRAM features

The *Low-latency RAM* (LLRAM) manager interface is a 256-bit AMBA® 5 AXI manager. The following table shows LLRAM AXI properties the Cortex®-R82 processor supports or requires the cluster interconnect and system to support.

Table 8-41: LLRAM features

AXI property	Supported by the Cortex®-R82 processor LLRAM	Interconnect support required
Multi_Copy_Atomicity	Yes	Yes
Ordered_Write_Observation	Yes	Yes
Atomic_Transactions	Yes Note: Yes if CFLRAMSHARED is HIGH and BROADCASTATOMIC is HIGH.	Yes
Poison	Yes	Yes
Check_Type	No	No
QoS_Accept	No	No
Trace_Signals	No	No
Loopback_Signals	No	No
Wakeup_Signals	Yes	Yes
Untranslated_Transactions	No	No
NSAccess_Identifiers	No	No
Max_Transaction_Bytes	Yes (64 bytes)	No

8.8.2 LLRAM attributes

The following table lists the read and write issuing capabilities for an eight-core Cortex®-R82 processor.

Table 8-42: LLRAM attributes

Attribute	Value	Comments
Write issuing capability	8	The maximum number of writes is 8. This value can be used by system components to size buffers when bridging to other interface protocols, for example PCIe. Normal Non-cacheable transactions can be removed from this limit by setting the control bit in the IMP_CLUSTERACTLR_EL1.
Read issuing capability	8	The maximum number of reads is 8.
Combined issuing capability	8	The combined issuing capability is 8.

Attribute	Value	Comments
Exclusive thread capability	Number of cores	Each hardware thread can have 1 exclusive access sequence in progress.
Write ID capability	8	The maximum write ID capability is 8. Only Device memory types with nGnRnE or nGnRE can have more than one outstanding transaction with the same AXI ID. All other memory types use a unique AXI ID for every outstanding transaction.
Write ID width	8	The ID encodes the source of the memory transaction. See the Encodings for AWIDL[7:0] table.
Read ID capability	8	The maximum read ID capability is 8.
Read ID width	8	The ID encodes the source of the memory transaction. See the Encodings for ARIDL[7:0] table.



Note

The issuing capability described here is the maximum possible for the whole cluster. This can be used to size interconnect capabilities if you want to achieve the maximum performance available. However, this maximum may not be reached depending on the Cortex®-R82 processor configuration and characteristics of your system.

Table 8-43: Encodings for AWIDL[7:0] and ARIDL[7:0]

Attribute	Value	Comments
Write ID	0bnnnmrrr	nnnn is the core number 0b0000-0b0111 in binary. If the nnnn is equal to the number of cores then it is for ACE-Lite Subordinate (ACELS) interface.
		m is 1 for read or write associated with atomics or exclusives. Otherwise m is 0.
		rrr is the internal buffer ID.
Read ID	0bnnnmrrr	nnnn is the core number 0b0000-0b0111 in binary. If the nnnn is larger or equal to the number of cores then it is for ACELS interface.
		m is 1 for read or write associated with atomics or exclusives and LLRAM Accelerator Coherency Port (LLRAM ACP) accesses. Otherwise m is 0.
		rrr is the internal buffer ID.



Note

These ID and transaction details are provided for information only. Arm strongly recommends that all interconnects and peripherals are designed to support any type and number of transactions on any ID to ensure compatibility with future products.

For more information about the AXI signals described in this manual, see the [AMBA® AXI Protocol Specification](#).

8.8.3 LLRAM transactions

The Cortex®-R82 processor does not generate bursts which cross a cache line boundary.

The cache linefill fetch length is always 64 bytes.

The Cortex®-R82 processor generates only a subset of all possible AXI transactions on the LLRAM interface.

The following points apply to AXI transactions that are generated by the core:

- Write transfers with none, or some byte strobes LOW can occur.
- Exclusive operations to LLRAM addresses that are marked as Outer Shareable generate exclusive transactions on the port only if CFGLLRAMSHARED is HIGH. Otherwise, they are executed internally.
- Exclusive operations to LLRAM addresses that are not marked as Outer Shareable are handled within the cluster and do not generate exclusive transactions on the LLRAM port.
- Atomic operations to LLRAM addresses that are marked as Outer Shareable generate atomic transactions on the port only if BROADCASTATOMICL and CFGLLRAMSHARED are HIGH. Otherwise, they are executed internally and do not generate atomic transactions (except when BROADCASTATOMICL is LOW and CFGLLRAMSHARED is HIGH, Outer Shareable atomics return SLVERR on the LLRAM port).
- Atomic operations to LLRAM addresses that are not marked as Outer Shareable are executed within the cluster and do not generate atomic transactions on the LLRAM port
- For cacheable memory, D-side accesses, the following accesses are possible:
 - Read transfers:
 - WRAP, 2 beats, 256-bit
 - Write transfers:
 - INCR, 1 beat, 256-bit, address aligned to 256-bits
 - INCR, 1 or 2 beats, 256-bit, address aligned to 512-bits
- For non-cacheable or device memory, the following accesses are possible:
 - Read transfers:
 - INCR, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit
 - Write transfers:
 - INCR, 1 beat, 256-bit, address aligned to 256-bits
 - INCR, 1 or 2 beats, 256-bit, address aligned to 512-bits
- For exclusives, the following accesses are possible:
 - Read exclusive transfers:
 - INCR, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bits
 - Write exclusive transfers:
 - INCR, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit

- For atomics, the following accesses are possible:
 - Write atomic transfers:
 - 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, 128-bit, or 256-bit

The following points apply to AXI transactions that are generated by the LLRAM ACP:

- Non-modifiable reads with zero length are not modified. The request on the LLRAM port is the same as the initial request received via the ACE-Lite Subordinate (ACELS) port.
- Reads and Writes to Normal memory that are larger than 512 bits, are split into two or more 512-bit transactions.
- Write transfers with none, or some or all byte strobes LOW can occur.
- For modifiable transfers, the following accesses are possible:
 - Read transfers:
 - INCR, 1 or 2 beats, 256-bit
 - Read transfers:
 - INCR or FIXED, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit
 - Write transfers:
 - INCR, 1 or 2 beats, 256-bit
- Non-modifiable transfers, the following accesses are possible:
 - Read transfers:
 - INCR or FIXED, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit
 - Write transfers:
 - INCR or FIXED, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit
- For exclusives, the following accesses are possible:
 - Read transfers:
 - INCR or FIXED, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit
 - Write transfers:
 - INCR or FIXED, 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, or 128-bit
- For atomics, the following accesses are possible:
 - 1 beat, 8-bit, 16-bit, 32-bit, 64-bit, 128-bit or 256-bit

8.8.4 Support for memory types

The Cortex®-R82 processor simplifies the coherency logic by downgrading some memory types.

The *Low-latency RAM* (LLRAM) port supports only Write-Through caching. LLRAM addresses that are marked as Write-Back cacheable are treated as Write-Through.

Accesses to the LLRAM port can only be cached in the L1 data caches and L1 instruction caches as determined by memory type and attributes. Accesses to the LLRAM port cannot be cached in the L2 cache.

Only the following memory types can be cached in the L1 data caches:

- Normal memory that is marked as both Inner Write-Through Cacheable and Outer Write-Through Cacheable.
- Normal memory that is marked as both Inner Write-Through Cacheable and Outer Write-Back Cacheable which is treated as Inner and Outer Write-Through.
- Normal memory that is marked as both Inner Write-Back Cacheable and Outer Write-Back Cacheable which is treated as Inner and Outer Write-Through.
- Normal memory that is marked as both Inner Write-Back Cacheable and Outer Write-Through Cacheable which is treated as Inner and Outer Write-Through.

All other Normal memory types are treated as Non-cacheable and are sent on the LLRAM port as Normal Non-cacheable.

Device memory is always treated as non-Gathering, non-Reordering, with no Early Write Acknowledgement (nGnRnE).

8.8.5 LLRAM write response

The *Low-latency RAM* (LLRAM) manager always accepts write responses without delay by holding BREADYL HIGH.

8.8.6 AXI4 compatibility mode

The Cortex®-R82 processor implements an AXI4 compatibility mode that enables you to use the Cortex®-R82 processor in a standalone environment where the AMBA® AXI5 interface is not required.

The *Low-latency RAM* (LLRAM) manager interface performs all atomics within the cluster at the shared level, rather than in the L1 memory system or external memory.

To enable AXI4 compatibility mode, you must ensure that the CFGLLRAMSHARED and RPOISONL signals are LOW.

While the LLRAM port also includes the WPOISONL output signal, this will never be driven HIGH by the Cortex®-R82 processor and can be safely left unconnected.

8.8.7 LLRAM privilege information

The *Low-latency RAM* (LLRAM) manager interface provides information to indicate whether the request is Privileged or Unprivileged on the ARPROTL[0] and AWPROTL[0] signals.

Where a request cannot be merged (Loads and Stores to Device memory, or Non-cacheable Store Exclusives) the ARPROTL[0] and AWPROTL[0] reflect the privilege state of the requestor. Where requests might have been merged, including all Non-cacheable and cacheable loads and stores, the ARPROTL[0] and AWPROTL[0] indicate that the request is privileged.

The LLRAM interface provides information about the Secure or Non-secure access on the ARPROTL[1] and AWPROTL[1] signals. The values of 0 indicates the access is Secure and the value of 1 indicates the access is Non-secure.

The LLRAM interface indicates whether a request is for a data or instruction fetch via the ARPROTL[2] and AWPROTL[2] signals.

8.9 MACP subordinate interface

The *Main Accelerator Coherency Port* (MACP) is a 128-bit subordinate interface that conforms to a subset of the ACE5-Lite specification. MACP is a single port attached to the shared L2 and therefore can access data that is shared between all cores in the Cortex®-R82 processor.

The MACP subordinate interface allows external agents to access memory through the *Main Manager* (MM) interface of the Cortex®-R82 processor. The MACP subordinate interface provides I/O coherency for external agents with the Cortex®-R82 processor L1 data and L2 caches (even if the L2 cache size parameter, `L2_CACHE_SIZE` is configured 0).

The `AXCACHE` values of transactions performed via the MACP interface control whether they look-up in the caches. Transactions that are marked as Device, Non-cacheable, or write=through (which is downgraded to Non-cacheable) do not look-up in the cache.

The MACP is optimized for cache line length accesses.

To maintain cache coherency, accesses are checked in all cached locations in the cluster. That is, the L1 data caches in each core and the L2 cache. L1 instruction caches are not checked for coherency because they should be coherent with the external memory.

The MACP supports stash requests from external agents. All stashing requests target the L2 cache and are always cacheline-sized.

8.9.1 MACP features

The Cortex®-R82 processor *Main Accelerator Coherency Port* (MACP) supports the following features.

Table 8-44: MACP features for the Cortex®-R82 processor

MACP property	Supported by the Cortex®-R82 processor
Port_Type	Accelerator
Continuous_Cache_Line_Read_Data	Yes
Multi_Copy_Atomicity	Yes
Ordered_Write_Observation	No
WriteEvict_Transaction	No
DVM_v8	No
Atomic_Transactions	Yes
DVM_v8.1	No
Cache_Stash_Transactions	Yes
DeAllocation_Transactions	No
Persistent_CMO	No
Poison	No
Check_Type	No
QoS_Accept	No
Trace_Signals	No
Loopback_Signals	No
Wakeup_Signals	Yes
Untranslated_Transactions	No
NSAccess_Identifiers	No

8.9.2 MACP attributes

For optimum performance, use the following guidelines for the *Main Accelerator Coherency Port* (MACP) transactions.

WriteUniquePtl transactions always incur a read-modify write sequence.

L2 resources are shared between the MACP and the cores. Therefore, heavy traffic on the MACP might, in some cases, reduce the performance of the cores.

Read transactions cannot allocate to the L1 cache, but use the Write-back Read-Allocate bits of the memory type (ARCAACHEA) to decide whether to allocate to the L2 cache.

Write transactions cannot allocate to the L1 cache, but use the Write-back Write-Allocate bits of the memory type (AWCACHEA) to decide whether to allocate to the L2 cache. Compared to

WriteUniqueFull, a WriteUniqueFullStash that targets allocation to the L2 cache is optimized to reduce the latency of a core read request that occurs shortly after the write request.

The following table describes the MACP attributes.

Table 8-45: MACP attributes

Attribute	Value	Description
Write acceptance capability	5	The MACP can accept up to five write transactions.
Read acceptance capability	5	The MACP can accept up to five read transactions.
Combined acceptance capability	6	The MACP can accept up to six transactions with: <ul style="list-style-type: none"> One read transaction and one write transaction Up to four read transactions or write transactions
Write ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.
Read ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.

8.9.3 MACP transaction types

The *Main Accelerator Coherency Port* (MACP) supports read and write transaction types with the same transfer size and length combinations.

The following table describes the MACP read and write transactions.



- x in the signal names is used to refer to both read and write signals. For example, AxLENA refers to both ARLENA and AWLENA signals.
- For all the write transactions,WSTRBA, any combination of bytes, including no bytes, are valid.

Table 8-46: Read and write transactions

Transaction	AxLENA	AxSIZEA	AxADDRA
1-byte INCR	0 (one beat)	0 (single byte)	-
2-byte INCR	0 (one beat)	1 (two bytes)	0b0 (Address aligned to 2-byte boundary)
4-byte INCR	0 (one beat)	2 (four bytes)	0b00 (Address aligned to 4-byte boundary)
8-byte INCR	0 (one beat)	3 (eight bytes)	0b000 (Address aligned to 8-byte boundary)
16-byte INCR	0 (one beat)	4 (16 bytes)	0b0000 (Address aligned to 16-byte boundary)

Transaction	AxLENA	AxSIZEA	AxADDRA
32-byte INCR	1 (two beats)	4 (16 bytes)	0b00000 (Address aligned to 32-byte boundary)
64-byte INCR	3 (four beats)	4 (16 bytes)	0b000000 (Address aligned to 64-byte boundary)

**Note**

The AMBA 5 ACE-Lite transaction types WriteUniqueFull and WriteUniquePtl were known in AMBA 4 ACE-Lite as WriteLineUnique and WriteUnique, respectively.

The following table lists the transaction types that are supported by the MACP:

Table 8-47: MACP supported transaction types

Transaction group	Transaction type
Read	ReadOnce
	ReadNoSnoop
Write	WriteUniquePtl
	WriteUniqueFull
	WriteUniqueFullStash
	WriteUniquePtlStash
	WriteNoSnoop
	StashOnceUnique
	StashOnceShared
Cache maintenance	CleanShared
	CleanInvalid
	MakeInvalid
Atomics	AtomicStore*
	AtomicLoad*
	AtomicSwap
	AtomicCompare

8.9.3.1 MACP transaction restrictions

The *Main Accelerator Coherency Port* (MACP) conforms to a subset of the ACE5-Lite specification.

The ACE5-Lite is described in the [AMBA® AXI Protocol Specification](#).

All transactions can be Secure or Non-secure.

All requests can specify Outer Shareable and Non-shareable using the AWDOMAINA and ARDOMAINA signals.

ARQOS and AWQOS signals are not present.

The Cortex®-R82 processor has the following additional restrictions on MACP. The requests that do not meet these restrictions generate a SLVERR response on RRESPA or BRESPA:

- Transactions with a length of more than one transfer are not supported when AxCACHEA[1] is 0.
- Exclusive accesses are not supported. ARLOCK and AWLOCK signals are not present.
- Barriers are not supported. The BRESPA response for any write transaction indicates global observability for the transaction.
- ARSIZEA and AWSIZEA signals are present. Only combinations of transaction address, burst length and burst size resulting in the following will be supported:
 - The total amount of data transferred is a power of 2, from one byte to a maximum of 64 bytes (one Cortex®-R82 processor cache line).
 - The burst address is aligned to the total amount of data transferred.
- The values of ARLENA and AWLENA are restricted to:

0	One beat
1	Two beats
3	Four beats
- ARBURST and AWBURST signals are present. Only a value of 0b01 (INCR) is supported.
- Only the following Atomic sizes and lengths are supported:
 - For AtomicStore*, AtomicLoad*, and AtomicSwap: 1 byte, 2 bytes, 4 bytes, and 8 bytes
 - For AtomicCompare: 2 bytes, 4 bytes, 8 bytes, 16 bytes, and 32 bytes

8.10 ACELS interface

The *ACE-Lite Subordinate* (ACELS) interface is a 128-bit ACE5-Lite subordinate interface that is shared between all cores in the Cortex®-R82 processor. It provides external access to the *Tightly Coupled Memories* (TCMs) and the *Low-latency RAM* (LLRAM) port.

The ACELS interface is used for two purposes:

- As a TCM subordinate enabling agents outside the cluster to access TCMs within the cores. The TCM subordinate also enables the cores within the Cortex®-R82 processor to access the TCMs of other cores through an interconnect loopback.
- As an LLRAM *Accelerator Coherency Port* (LLRAM ACP) enabling coherent external access to the LLRAM port.

Both of these functions are provided by the same physical port, ACELS, that responds to two different ranges of address, one for the TCM subordinate and one for the LLRAM ACP. However, they are described as separate concepts in this chapter.

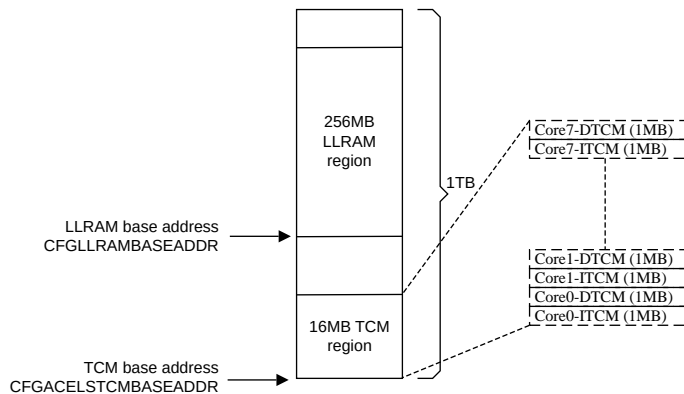
The Cortex®-R82 processor TCM subordinate and LLRAM ACP address regions must not overlap. The Cortex®-R82 processor ACELS interface generates SLVERR error response to access requests that fall within both the TCMs and the LLRAM ACP addresses.

The ACELS interface uses base addresses for routing the access to the 16MB TCM region and 256MB LLRAM ACP region:

- The 16MB TCM region is divided into eight 2MB blocks, one for each core. TCMs (ITCM and DTCM) within a core are mapped into two 1MB regions.
- The LLRAM ACP address region is the same as the LLRAM address region visible to the cores within the Cortex®-R82 processor.

The following figure shows how each TCM and the LLRAM ACP are mapped into the ACELS interface address space.

Figure 8-5: ACELS memory map



The ACELS memory map is unchanged irrespective of whether the Cortex®-R82 processor is configured to operate in Split-mode, Lock-mode, or Hybrid-mode. However, when operating in Lock-mode, the TCMs must only be accessed by addressing the physical cores. This means that accesses can only be made to even-numbered cores in the TCM region and accesses to the redundant cores (which are odd-numbered) will return a decode error without forwarding the request to the TCMs.

The ACELS port is typically connected to an interconnect, through which various system agents can access the TCMs and LLRAM port.

Even though the TCM subordinate and LLRAM ACP share the ACELS port, they operate largely independently. The ACELS port supports out-of-order completion between accesses to each core's TCM subordinates and accesses to the LLRAM port. The ACELS transactions with different ACE IDs are able to complete out-of-order.

For any request that is targeting the TCM subordinate and LLRAM ACP, the total amount of data transferred is a power of 2 and can be greater than 64 bytes that is one Cortex®-R82 processor cache line.

The Cortex®-R82 processor ACELS port does not create ordering dependencies between the ACELS reads and writes, or between the ACELS reads with different IDs, or between the ACELS writes with different IDs.

The Cortex®-R82 processor ACELS AXI ID signals are 8 to 24 bits wide and configured with the `ACELS_ID_WIDTH` parameter. In order for an AXI or ACE-Lite manager with an ACE ID width larger than the Cortex®-R82 processor ACELS AXI ID width to access the ACELS port, that manager must be connected through logic that compresses the AXI ID signals. The Cortex®-R82 processor does not provide such logic.

The ACELS port contains enough buffering to support 128-bit per cycle sustained transfers to/from TCM or LLRAM port memory with no wait state. However, the ACELS port might be unable to immediately accept a transaction request and might instead apply back-pressure on the port when many transactions to TCM or LLRAM port memory with wait states are outstanding.

The ACELS port includes ARLOCK and AWLOCK pins, but only supports single beat exclusives to the LLRAM ACP. Exclusives targeting the LLRAM ACP with AXLENS not set to 0 are not supported and will return a subordinate error. Furthermore, exclusives are not supported to the TCM subordinate and all exclusive requests targeting the TCM subordinates will not return an EXOKAY response and will therefore fail the exclusive access.

The ACELS port includes the AWATOP pins to ease integration with an AXI or ACE manager that includes atomic support. The ACELS port only supports atomics for the LLRAM ACP. Any atomic request targeting the TCM subordinate will return a SLVERR response.

For atomics with a load component (AtomicLoad, AtomicStore and AtomicCompare), the read response and write response are not guaranteed to be consistent. For example, the read response might return OKAY if the read completed successfully while the write might return an error response if the write component fails. It is recommended the manager connected to the ACELS port consumes both the read and write response in order to determine if the atomic has completed successfully.

8.10.1 ACELS features

The Cortex®-R82 processor *ACE-Lite Subordinate* (ACELS) interface supports the following features.

Table 8-48: ACELS features for the Cortex®-R82 processor

ACELS property	Supported by the Cortex®-R82 processor
Port_Type	Accelerator
Continuous_Cache_Line_Read_Data	Yes
Multi_Copy_Atomicity	Yes
Ordered_Write_Observation	No
WriteEvict_Transaction	No
DVM_v8	No

ACELS property	Supported by the Cortex®-R82 processor
Atomic_Transactions	Yes Note: Atomics are supported for LLRAM ACP but not for the TCM subordinate. Atomic requests targeting TCM subordinate will return a SLVERR response.
DVM_v8.1	No
Cache_Stash_Transactions	No
DeAllocation_Transactions	No
Persistent_CMO	No
Poison	No
Check_Type	No
QoS_Accept	No
Trace_Signals	No
Loopback_Signals	No
Wakeup_Signals	Yes
Untranslated_Transactions	No
NSAccess_Identifiers	No

8.10.2 ACELS attributes

For optimum performance, use the following guidelines for *ACE-Lite Subordinate* (ACELS) interface transactions.

WriteUniquePtl transactions are sent out on the ACE-Lite bus.

Some L1 and *LLRAM Coherency Unit* (LCU) memory system resources are shared between the ACELS interface and the cores. Therefore, heavy traffic on the ACELS interface might, in some cases, reduce the performance of the cores.

The following table describes the ACELS attributes where NUM_CORES is the number of logical cores from 1 to 8.

Table 8-49: ACELS attributes

Attribute	Value	Description
Write acceptance capability	$(8 * \text{NUM_CORES}) + 10$	The write acceptance capability is dependent on the number of cores. For example, an MP4 configuration has a theoretical acceptance capability of 42 write transactions.
Read acceptance capability	$(7 * \text{NUM_CORES}) + 10$	The read acceptance capability is dependent on the number of cores. For example, an MP4 configuration has a theoretical acceptance capability of 38 read transactions.

Attribute	Value	Description
Combined acceptance capability	Write acceptance capability + Read acceptance capability - (4 * NUM_CORES)	The combined acceptance capability is dependent on the number of cores. For example, an MP4 configuration has a theoretical acceptance capability of 44 transactions.
Write ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.
Read ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.

8.10.3 ACELS transaction types

The *ACE-Lite Subordinate* (ACELS) interface conforms to the ACE5-Lite specification.

The Cortex®-R82 processor ACELS interface supports the majority of ACE5-Lite burst types and lengths. See [8.10.3.1 ACELS transaction restrictions](#) on page 171 for the restrictions that apply to the ACELS interface.

For more information on the ACE5-Lite, see the [AMBA® AXI Protocol Specification](#).

The following table lists the transaction types that are supported by the ACELS.

Table 8-50: ACELS supported transaction types

Transaction group	Transaction type
Read	ReadOnce
	ReadNoSnoop
Write	WriteUniquePtl
	WriteUniqueFull
	WriteNoSnoop
Atomics Note: Atomics are supported for LLRAM ACP but not for the TCM subordinate. Atomic requests targeting TCM subordinate will return a SLVERR response.	AtomicStore
	AtomicLoad
	AtomicSwap
	AtomicCompare

8.10.3.1 ACELS transaction restrictions

The following transactions cause SLVERR or DECERR error on the *ACE-Lite Subordinate* (ACELS) interface.

The following cases generate a SLVERR response on RRESPS or BRESPS:

- Non-modifiable bursts (ARCACHES[1] or AWCACHES[1] is set to 0) when ARLENS > 0 or AWLENS > 0.

- ARSNOOPS is non-zero, or AWSNOOPS[3:1] is non-zero.
- Non-secure access to the TCM subordinate when ARPROTS[1] or AWPROTS[1] is set to 1.
- Atomic requests to the TCM subordinate.
- Non-single-beat exclusive requests to the LLRAM ACP (ARLENS > 0)
- IMP_CLUSTERACELSCTLR_EL1[x] is set to 0, where x is the (core number * 2) that the accessed TCM belongs to.
- IMP_CLUSTERACELSCTLR_EL1[x] is set to 0 and ARPROTS[0] or AWPROTS[0] is set to 0, where x is the (core number * 2) + 1 that the accessed TCM belongs to.
- TCM subordinate access to a powered off core.
- Double bit error on a read to the *Tightly Coupled Memories* (TCMs).
- SLVERR response from the *Low-latency RAM* (LLRAM) interface as a response to an access from the LLRAM ACP.

The following cases generate a DECERR response on RRESPS or BRESPS:

- Access to the region of ACELS memory map if the address falls outside both the LLRAM and TCM regions.
- Access to the region of ACELS memory map if the address falls into both the LLRAM and TCM regions.
- Access to the gap in ACELS memory map between TCMs.
- DECERR response from the LLRAM interface as a response to an access from the LLRAM ACP.
- Access to the TCM region of a non-existent core.
- Access to the TCM region of a redundant core when operating in Lock-mode.

The following cases generate an OK response on RRESPS or BRESPS:

- An exclusive (locked) read access to the TCM region, except where a SLVERR or DECERR occurs.
- An exclusive (locked) write access to the TCM region, except where a SLVERR or DECERR occurs.

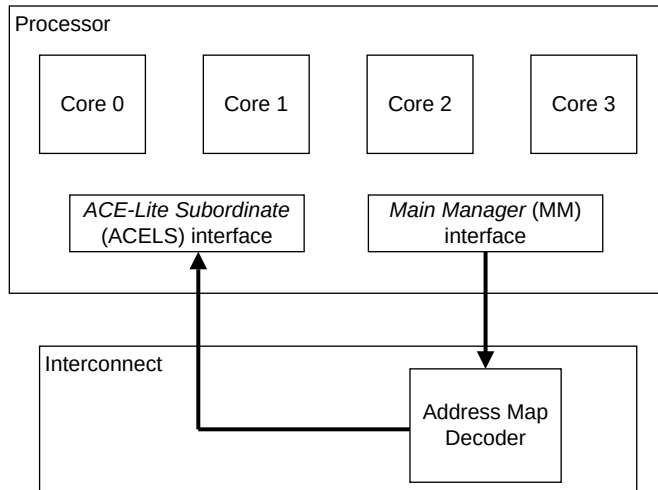
8.10.4 TCM subordinate

The Cortex®-R82 processor TCM subordinate enables agents outside the cluster to access all the TCMs within the Cortex®-R82 cluster. It also enables any core within the Cortex®-R82 processor to access all the TCMs within the cluster, except its own, via an interconnect loopback, that is assuming the *ACE-Lite Subordinate* (ACELS) port is connected to the interconnect and the Cortex®-R82 processor is able to access the ACELS port through regular *Main Manager* (MM) memory accesses to the address region starting at CFGACELSTCMBASEADDR. The software running on any given core within the Cortex®-R82 processor shall not access its own TCMs via

the TCM subordinate. It shall use the private addresses defined by CFGITCMBASEADDRm or CFGDTCMBASEADDRm to access its own TCMs.

The following figure shows an example of a loopback address mapping for a core access to the ACELS through the interconnect. It is recommended that the Main Manager interface is used for loopback to the TCMs via TCM subordinate, and not the LLRAM, LLPP or SPP interfaces.

Figure 8-6: Loopback address mapping



The Cortex®-R82 processor TCM subordinate is capable of sustaining 128 bits per cycle of:

- Read bandwidth when reading from any number of TCMs with no wait states. This assumes no contention for accessing the TCMs.
- Write bandwidth when writing to any number of TCMs with no wait states. This assumes no contention for accessing the TCMs.

As the TCMs are banked for efficient sharing between the core and the TCM subordinate, the Cortex®-R82 processor TCM subordinate bandwidth tends towards 128 bits per cycle when both a core and the TCM subordinate access the same TCM with no wait state sequentially.

The Cortex®-R82 processor TCM subordinate address space is a 16MB region located at a 16MB-aligned base address set by the configuration input signal CFGACELSTCMBASEADDR[PA_W-1:24], providing access to all TCM memories within the processor. The base address set by the configuration input signal CFGACELSTCMBASEADDR[PA_W-1:24] must match the address the ACELS port in the system memory map.

The 16MB region is divided into eight 2MB blocks, one for each core. Each 2MB block is subdivided into two 1MB regions. TCMs (ITCM and DTCM) within a core are mapped into two 1MB regions. The lower 1MB is mapped to the ITCM and the upper 1MB to the DTCM. If the size of any TCM is less than the maximum 1MB, the remaining upper part will be inaccessible and return a DECERR response.

The memory map is the same regardless of the number of cores configured, and the sizes and number of TCMs present. Regions associated with cores that are not present generate a DECERR response. If a TCM is accessed outside of its configured range, the TCM subordinate generates a DECERR error response. Such accesses include addresses that map to cores which have not been implemented, addresses which do not map to any TCM, and addresses that map to a TCM but are too high for the implemented size of that TCM.

For each core, access control checks can be enabled for the TCM subordinate transactions by programming `IMP_CLUSTERACELSCCLR.TCMACCLVL`. Access control allows either all transactions or only privileged transactions to access the TCM. If a transaction is not permitted, the TCM subordinate generates a SLVERR response.

For transaction requests that target cores which have been powered down, the TCM subordinate generates a SLVERR error response.

If the Cortex®-R82 processor TCM subordinate accesses TCMs in a core which is in WFI or WFE low-power state but not in a retention or power down state, then the access proceeds. In this case, the clocks to the core are re-enabled if necessary.

8.10.4.1 Accessing TCMs configured with ECC

When a *Tightly Coupled Memory* (TCM) implements *Error Correcting Code* (ECC), the ECC is generated within the core before writing to the TCM.

If the core detects an error on reads or writes the behavior is:

Writes

- If a write to TCM requires a read-modify-write, and a correctable error is detected when reading the TCM, the ECC is recalculated.
- If a write to TCM requires a read-modify-write, and a non-correctable error is detected when reading the TCM, a SLVERR response is returned.

Reads

- For a correctable error, the core corrects the data and returns corrected data with OKAY response.
- For a non-correctable error, a SLVERR response is returned.

8.10.4.2 TCM subordinate attributes

This section describes the capabilities and attributes of the TCM subordinate interface.

The TCM subordinate interface does not support:

- Exclusive transactions, therefore, `AmLOCK` is not used.
- Data and instruction transaction signaling, therefore, `AmPROT[2]` is not used.
- QoS is not supported, therefore, `AmQOS` is not used.

- Multiple address region signaling is not supported, therefore, AmREGION is not used.

The following table shows the TCM subordinate interface attributes where NUM_CORES is the number of logical cores from 1 to 8.

Table 8-51: TCM subordinate interface attributes

Attribute	Value	Comments
Write acceptance capability	NUM_CORES * 8	The maximum number of outstanding write transactions that a subordinate can accept.
Read acceptance capability	NUM_CORES * 7	The maximum number of outstanding read transactions that a subordinate can accept.
Combined acceptance capability	NUM_CORES * 15	The maximum number of outstanding transactions that a subordinate can accept.
Write ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.
Read ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.

8.10.5 LLRAM ACP

The *Low-latency RAM Accelerator Coherency Port* (LLRAM ACP) enables agents outside the cluster to access the LLRAM port coherently. The *ACE-Lite Subordinate* (ACELS) interface routes accesses to addresses within the LLRAM ACP address region to the LLRAM ACP. The LLRAM ACP address region is the same as the LLRAM address region visible to the cores within the Cortex®-R82 processor.

The Cortex®-R82 processor includes coherency hardware that automatically manages the contents of the L1 data caches within the cluster to ensure all cores within the cluster and uncached agents connected to the LLRAM ACP have a coherent view of LLRAM addresses.

The LLRAM ACP is capable of sustaining 128 bits per cycle of:

- Read bandwidth when reading from the LLRAM. This assumes no contention for accessing the LLRAM.
- Write bandwidth when writing to the LLRAM. This assumes no contention for accessing the LLRAM.

The LLRAM ACP interface provides information to indicate whether the request is Privileged or Unprivileged on the ARPROTL[0] and AWPROTL[0] signals.



- A Cacheable transaction from the Cortex®-R82 processor or ACELS interface is always indicated as Privileged (ARPROTL[0] = 1 and AWPROTL[0] = 1).
- A Non-cacheable or Device transaction from the ACELS interface has the incoming ARPROTS[0] and AWPROTS[0] values.

The LLRAM ACP provides information about the security attributes of accesses on the ARPROTL[1] and AWPROTL[1] signals. The value of 0 indicates the access is Secure and the value of 1 indicates the access is Non-secure.

The LLRAM ACP interface indicates whether a request is for a data or instruction fetch via the ARPROTL[2] and AWPROTL[2] signals.

8.10.5.1 LLRAM ACP attributes

This section describes the capabilities and attributes of the Cortex®-R82 processor *Low-latency RAM Accelerator Coherency Port* (LLRAM ACP).

The Cortex®-R82 processor LLRAM ACP does not support:

- QoS is not supported, therefore, AmQOS is not used.
- Multiple address region signaling is not supported, therefore, AmREGION is not used.
- Barriers are not supported. The write response for any write request indicates global observation of that write.

The following table shows the Cortex®-R82 processor LLRAM ACP attributes.

Table 8-52: LLRAM ACP attributes

Attribute	Value	Comments
Write acceptance capability	12	The maximum number of outstanding write transactions that a subordinate can accept.
Read acceptance capability	12	The maximum number of outstanding read transactions that a subordinate can accept.
Combined acceptance capability	14	The maximum number of outstanding read and write transactions that a subordinate can accept.
Write ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.
Read ID width	8 to 24 bits	Unused bits tied to zero if fewer bits required.

8.11 Utility bus

The Utility bus provides access to control registers for various system components in the Cortex®-R82 processor. The Utility bus is implemented as a 64-bit AMBA® 5 AXI subordinate port.

The Utility bus provides memory-mapped access to the following register families:

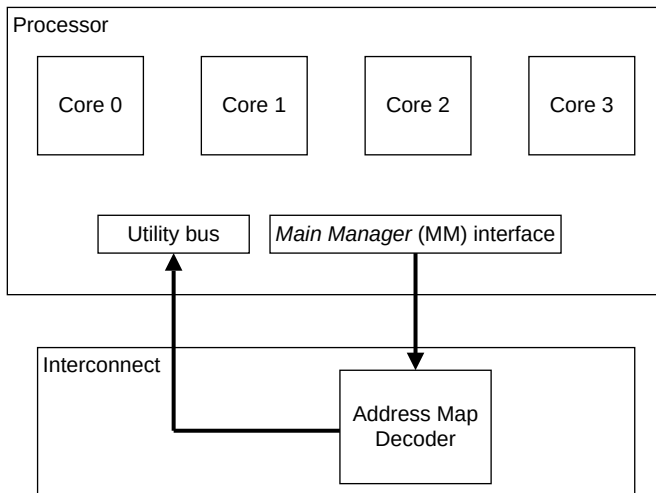
- Per-core and the cluster *Power Policy Units* (PPUs) registers.
- Per-core *Reliability, Availability, and Serviceability* (RAS) registers.

If your system has a *System Control Processor* (SCP), the Utility bus enables the SCP to manage the power policy and the error handling of the Cortex®-R82 processor.

If your system does not have an SCP or the SCP is limited in its scope, you can connect the Utility bus to the system interconnect. A core within the Cortex®-R82 processor, then, can access all the PPU and the RAS registers through the *Main Manager* (MM) port which is connected via a loopback to the Utility bus.

The following figure shows an example of a loopback address mapping for a core access to the Utility bus through the interconnect.

Figure 8-7: Loopback address mapping



Note

The Cortex®-R82 processor supports the loopback address mapping between the following ports:

- The MM port to the Utility bus.
- The *Shared Peripheral Port* (SPP) to the Utility bus.
- The MM port to the *ACE-Lite Subordinate* (ACELS) port.

8.11.1 Utility bus accesses

Transactions on the Utility bus comply with a subset of the AXI 5 bus protocol. Accesses must be either 32-bits or 64-bits. Any other sized access generates a SLVERR response from the Utility bus.

You must observe the following requirements when accessing Utility bus:

- Only ReadNoSnoop and WriteNoSnoop transaction types are supported.
- Only 32-bit accesses or 64-bit accesses are supported. Therefore, ARSIZEU or AWSIZEU must be either 0b010 for 32-bit sized accesses, or 0b011 for 64-bit sized accesses. Any other access size generates a SLVERR response from the Utility bus.
- Only single beat bursts are supported. Therefore, ARLENU or AWLENU must be 0b00000000. Any other burst length generates a SLVERR response from the Utility bus.
- All system components control registers only support Secure accesses and data accesses on the Utility bus. Any accesses to these registers with the Non-secure bit set generate a SLVERR response.

- No exclusives supported. The Utility bus treats them as plain accesses and does not abort them.
- No atomics supported.

Arm recommends the following when accessing the Utility bus:

- ARCACHEU or AWCACHEU is either 0b0000 or 0b0001, although other values are accepted and ignored.
- ARBURSTU or AWBURSTU is 0b01, although other values are accepted and ignored.
- ARLOCKU or AWLOCKU is tied LOW, as there is no exclusive monitor present.

The following table describes the Utility bus attributes.

Table 8-53: Utility bus attributes

Attribute	Value	Description
Write acceptance capability	1	The Utility bus can accept one write transaction.
Read acceptance capability	1	The Utility bus can accept one read transaction.
Combined acceptance capability	2	The Utility bus can accept up to two transactions.
Write ID width	1 to 24 bits	Unused bits tied to zero if fewer bits required.
Read ID width	1 to 24 bits	Unused bits tied to zero if fewer bits required.

8.11.2 Base addresses for system components

Each set of system registers is grouped on separate 64KB page boundaries allowing access control to be enforced by the memory management. An alternative 4kB register spacing can be selected using the `DENSE_CS_ADDR_MAP` parameter.

See [B.1.2.2.3 PPU_PWSR, Power Status Register](#) on page 1481 for information on the base addresses for each set of system component registers that the external agents can access using the Utility bus.

8.12 Direct access to internal memories

The Cortex®-R82 processor provides a mechanism to read the internal memories that are used by the L1 and L2 caches and *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs and TLB structures through **IMPLEMENTATION DEFINED** System registers.

This functionality enables direct reading of cache RAMs by software and can be useful when investigating issues where the coherency between the data in the cache and data in the system memory is broken.

8.12.1 Direct access to L1 memory

The Cortex®-R82 processor provides a mechanism to read the internal memories that are used by the L1 caches and TLB structures through **IMPLEMENTATION DEFINED** System registers.

The appropriate memory block and location are selected using one of several system instructions. The data is read from read-only registers after performing the appropriate Read Operation system instruction. These operations are available both in EL1 and EL2 but EL1 accesses can be trapped using ACTLR_EL2.CDBG. In EL0, executing these instructions results in an Undefined Instruction exception.

The following table shows the system registers and system instructions to access L1 memory.

Table 8-54: System registers and system instructions used to access L1 memory

Register name	Function	Access	Operation	Register Data
IMP_CDBGDR0_EL1	Cache Debug Data Register 0	Read-only	MRS <Xt>, S3_2_c15_c0_0	Data
IMP_CDBGDR1_EL1	Cache Debug Data Register 1	Read-only	MRS <Xt>, S3_2_c15_c0_1	Data
SYS IMP_CDBGDCT	L1 Data Cache Tag Read Operation	System instruction	SYS S1_2_c15_c2_0, <Xt>	Set/Way
SYS IMP_CDBGICT	L1 Instruction Cache Tag Read Operation	System instruction	SYS S1_2_c15_c2_1, <Xt>	Set/Way
SYS IMP_CDBGTT	TLB Tag Read Operation	System instruction	SYS S1_2_c15_c2_2, <Xt>	Index/Way
SYS IMP_CBGDCD	L1 Data Cache Data Read Operation	System instruction	SYS S1_2_c15_c4_0, <Xt>	Set/Way/Offset
SYS IMP_CBGICD	L1 Instruction Cache Data Read Operation	System instruction	SYS S1_2_c15_c4_1, <Xt>	Set/Way/Offset
SYS IMP_CBGTD	TLB Data Read Operation	System instruction	SYS S1_2_c15_c4_2, <Xt>	Index/Way

Execution of one of the SYS IMP_CDBGDCT, SYS IMP_CDBGICT, SYS IMP_CBGDCD or SYS IMP_CBGICD operations must be followed by a read of IMP_CDBGDR0_EL1 register, and if necessary by a subsequent read of IMP_CDBGDR1_EL1 register. Similarly, a read from IMP_CDBGDR0_EL1 or IMP_CDBGDR1_EL1 registers must have been preceded by execution of one of the SYS IMP_CDBGDCT, SYS IMP_CDBGICT, SYS IMP_CBGDCD or SYS IMP_CBGICD operations.

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

8.12.2 Direct access to L2 and LCU memory

The Cortex®-R82 processor provides a mechanism to read the internal memories that are used by the L2 cache and the *LLRAM Coherency Unit* (LCU) and TLB structures through **IMPLEMENTATION DEFINED** System registers.

The appropriate memory block and location are selected using one of several system instructions. The data is read from read-only registers after performing the appropriate Read Operation system instruction. These operations are available both in EL1 and EL2 but EL1 accesses can be trapped using ACTLR_EL2.CDBG. In EL0, executing these instructions results in an Undefined Instruction exception.

The following table shows the system registers and system instructions to access L2 and LCU memories.

Table 8-55: System registers and system instructions used to access L2 and LCU memories

Register name	Function	Access	Operation	Register Data
IMP_CLUSTERCDBGDR0_EL1	Cluster Cache Debug Data Register 0	Read-only	MRS <Xt>, S3_2_c15_c3_0	Data
SYS IMP_CLUSTERCDBGL2D	L2 Cache Data Read Operation	System instruction	SYS S1_2_c15_c4_3, <Xt>	Set/Way
SYS IMP_CLUSTERCDBGL2DT	L2 Cache Duplicate L1 Tag Read Operation	System instruction	SYS S1_2_c15_c3_3, <Xt>	Set/Way
SYS IMP_CLUSTERCDBGL2T	L2 Cache Tag Read Operation	System instruction	SYS S1_2_c15_c2_3, <Xt>	Index/Way
SYS IMP_CLUSTERCDBGLCUdT	LCU Duplicate L1 Tag Read Operation	System instruction	SYS S1_2_c15_c3_4, <Xt>	Set/Way/Offset

Execution of one of the SYS IMP_CLUSTERCDBGL2D, SYS IMP_CLUSTERCDBGL2DT, SYS IMP_CLUSTERCDBGL2T or SYS IMP_CLUSTERCDBGLCUdT operations must be followed by a read of IMP_CLUSTERCDBGDR0_EL1 register. Similarly, a read from IMP_CLUSTERCDBGDR0_EL1 register must have been preceded by execution of one of the SYS IMP_CLUSTERCDBGL2D, SYS IMP_CLUSTERCDBGL2DT, SYS IMP_CLUSTERCDBGL2T or SYS IMP_CLUSTERCDBGLCUdT operations.

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

8.13 Exclusives and atomics support

The Cortex®-R82 processor supports load/store exclusive and atomic instructions on some interfaces and not on others.

The Cortex®-R82 processor does not support exclusives and atomics for the following cases:

- Software running from a core within the Cortex®-R82 processor executing load/store exclusive or atomic instructions accessing the *Low-latency Peripheral Port* (LLPP) region. A synchronous External abort is taken if software executes an exclusive or atomic targeting the LLPP.
- Software running from a core within the Cortex®-R82 processor executing load/store exclusive or atomic instructions accessing the *Shared Peripheral Port* (SPP) region. A synchronous External abort is taken if software executes an exclusive or atomic targeting the SPP.
- Incoming atomic or exclusive transactions on the TCM subordinate implemented by the *ACE-Lite Subordinate* (ACELS) interface. A SLVERR response is returned if software executes an atomic request targeting the TCM subordinate. An OK response is returned if software executes an exclusive request targeting the TCM subordinate, failing any exclusive requests.
- Incoming non-single-beat exclusive transactions on the *LLRAM Accelerator Coherency Port* (LLRAM ACP) implemented as ACE-Lite Subordinate (ACELS) interface. A SLVERR response is returned if software executes an exclusive request with ARLENS>0 targeting the LLRAM ACP.

- Incoming cacheable atomic or exclusive transactions on the *Main Accelerator Coherency Port* (MACP). A SLVERR response is returned if software executes a cacheable atomic request targeting the MACP but Non-cacheable atomics are supported. The MACP cannot receive exclusive requests (AxLock is not implemented) and can therefore only return an OK response, failing any exclusive requests.
- Incoming atomic or exclusive transactions on the Utility bus. The Utility bus cannot receive exclusive requests (AxLock is not implemented) and can therefore only return an OK response, failing any exclusive requests. The Utility bus cannot receive atomic requests (AWATOP is not implemented) and therefore cannot receive atomic requests. It is the responsibility of the system to abort any atomics targeting the Utility bus.

You can build systems that do not require support for exclusives or atomics outside the Cortex®-R82 processor. However, if your system requires support for exclusives outside the Cortex®-R82 processor, the system must implement a global exclusive access monitor. The global access monitor must signal to the Cortex®-R82 processor through the EVENTIREQ input when it is cleared.

The following table describes the support for exclusives and atomics for the following cases:

- Software running from a core within the Cortex®-R82 processor executing load/store exclusive or atomic instructions accessing the *Low-latency RAM* (LLRAM) region.
- Incoming transactions on the *LLRAM Accelerator Coherency Port* (LLRAM ACP) implemented as *ACE-Lite Subordinate* (ACELS) interface.



Note

- LLRAM ACP System Shareable transactions are treated as Outer Shareable.
- X in the following table means that the value can be either HIGH or LOW.

Table 8-56: Exclusives and atomics support for LLRAM and LLRAM ACP

Type	Shareability	CFGLLRAMSHARED	BROADCASTATOMICL	Access to LLRAM region	Incoming LLRAM ACP transaction
Exclusives	Non-shareable	X	X	Executed in cluster	
	Inner Shareable	X	X	Executed in cluster	
	Outer Shareable	HIGH	X	Output to LLRAM	
		LOW	X	Executed in cluster	
Atomics	Non-shareable	X	HIGH	Executed in cluster	
		X	LOW	Executed in cluster	
	Inner Shareable	X	HIGH	Executed in cluster	
		X	LOW	Executed in cluster	
	Outer Shareable	HIGH	HIGH	Output to LLRAM	
		HIGH	LOW	SLVERR	
		LOW	HIGH	Executed in cluster	
		LOW	LOW	Executed in cluster	

The following table describes the support for exclusives and atomics for the following cases:

- Software running from a core within the Cortex®-R82 processor executing load/store exclusive or atomic instructions accessing the *Main Manager* (MM) region.
- Incoming transactions on the *Main Accelerator Coherency Port* (MACP).

**Note**

X in the following table means that the value does not have any effect on the exclusives or atomics.

Table 8-57: Exclusives and atomics support for MM and MACP

Type	Shareability	BROADCAST signals		Atomic	Access to MM region	Incoming MACP transaction
		OUTERM	ATOMICM			
Exclusives	Non-shareable	X	X	X	Executed in cluster	SLVERR
	Inner/ Outer Shareable	HIGH	X	X	Output to MM, if Non-cacheable or Device. Otherwise depends on which cache holds the data.	SLVERR
		LOW	X	X	Output to MM, if Non-cacheable or Device. Otherwise executed in cluster.	SLVERR
Atomics	X	X	X	Near	Executed in cluster	Cannot happen. Atomics from MACP always treated as far.
		X	HIGH	Far	Output to MM	Forward to MM Note: Non-cacheable atomics are supported. Cacheable atomic transactions return SLVERR.
		X	LOW	Far	SLVERR	SLVERR

8.14 Real-time considerations

The Cortex®-R82 processor provides mechanisms and registers to ensure that your system can meet various real-time requirements.

This section describes the conditions, mechanisms, and register controls for the Cortex®-R82 processor so that your system can achieve:

- Bounded interrupt latency response under the best-case and worst-case conditions
- Hierarchical real-time requirements on interfaces and memories
- Quality of service capabilities to prioritize a core or a complete cluster.



The `DC ISW`, `DC CSW` and `DC CISW` instructions for the L1 data cache are serialized through the L2 cache. To ensure that *Low-latency RAM* (LLRAM) accesses cached through the L1 caches have higher real-time response than the *Main Manager* (MM) accesses, Arm recommends that you do not use these instructions for critical software sections.

8.14.1 Interrupt latency

The Cortex®-R82 processor guarantees a bounded interrupt latency response provided that your system and software meet certain conditions.

The Cortex®-R82 processor has an interrupt latency of 60 SCLK cycles under best-case interrupt latency conditions and 120 SCLK cycles under worst-case interrupt latency conditions.

Interrupt latency refers to the number of SCLK cycles from the assertion of a *Shared Peripheral Interrupt* (SPI) pin to the cycle in which the instruction before the first non-generic instruction in the interrupt handler retires. This includes the time for the execution of the instructions for identifying the interrupt, branching to the specific interrupt handler, stacking the relevant registers, and clearing the interrupt mask.

This section describes the conditions that your system and software should fulfill for the Cortex®-R82 processor to achieve the best-case and worst-case interrupt latency cycles.

8.14.1.1 Interrupt handler

The interrupt handler code can greatly affect the interrupt latency of your system.

The Cortex®-R82 processor accepts and processes interrupts within 60 and 120 SCLK cycles, if your interrupt handler meets the following conditions:

- The interrupt is handled by the target Exception level. In other words, the interrupt is not trapped by the hypervisor to route it to a virtual machine which can handle the interrupt.
- The related exception vector and interrupt handler code belong to *Memory Protection Unit* (MPU) regions of 4KB or larger.
- There is a first, generic part of the handler, that is common for all interrupts. This part is written in Assembly. The code is optimized to take advantage of the processor multi-issuing capabilities.
- There is a second, non-generic part of the handler, that depends on the interrupt ID. This part may be written either in Assembly or a higher-level language, such as C.
- The generic handler stacks general-purpose registers. The generic handler does not stack NEON/FP registers. It is assumed that the non-generic handler does not use NEON/FP instructions.
- The generic handler acknowledges the highest-priority interrupt and reads its interrupt ID.
- The generic handler re-enables (unmasks) interrupts as soon as possible, so that higher-priority interrupts can be taken.

- The generic handler looks up the interrupt ID from a table in memory and finds out which non-generic handler function must be called.
- The generic handler calls the non-generic handler to service the specific interrupt.
- The generic handler is executed from the *Instruction Tightly Coupled Memory* (ITCM).
- The generic handler uses the handler-specific stack which is in the *Data Tightly Coupled Memory* (DTCM).
- The generic handler does not perform any access to the *Main Manager* (MM) port, *Low-latency RAM* (LLRAM) port, *Low-latency Peripheral Port* (LLPP), and *Shared Peripheral Port* (SPP).
- The generic handler does not perform any exclusive or atomic memory access.

An example of such a generic handler for EL1 IRQs, from the vector entry up to the function call of the non-generic handler, is given in [D.1 Generic handler example, part 1](#) on page 2197.

An example of the remainder of the generic handler, from the return of the non-generic handler function up to the exception return is given in [D.2 Generic handler example, part 2](#) on page 2198.

Using this example handler as a reference, the best-case and worst-case interrupt latency is measured from the cycle when the SPI IRQ pin of the GIC is asserted, up to the cycle when the first instruction of the function called by the BLX X3 instruction has retired.

8.14.1.2 Best-case interrupt latency conditions

Under best-case interrupt latency conditions, the interrupt is accepted as soon as it is received by the core within the Cortex®-R82 processor. There cannot be ongoing interrupts or operations such as Device, atomic, or exclusive operations that would delay the interrupt handling.

The Cortex®-R82 processor accepts and processes the interrupt within 60 SCLK cycles if your system and software meet the following conditions:

- The *Generic Interrupt Controller* (GIC) distributor used is an Arm product supporting real-time interrupts, such as GIC-625
- The GIC distributor is clocked at half the frequency as the Cortex®-R82 cluster, with no bus bridges between the two.
- The GIC redistributor has enabled combined packets (GICR_FCTLR.ECP field is set to 1).
- The *Shared Peripheral Interrupt* (SPI) which is asserted is a low-latency targeted SPI (the related GICD_IROUTER<n>.IRM field is set to 0).
- Interrupt configuration (routing, priority, group assignment, per-interrupt enables, group enables, exception vector) is static.
- The level interrupts are not deasserted before they are serviced.
- Any number of SPIs are allowed to be asserted at the same time for the same core with any priority structure. Interrupts are not delivered to any other cores in the cluster.
- Data memory barrier instructions and instructions that write to memory with release semantics do not delay interrupts. That is, either of the following applies:

- An ELO or EL1 source context takes an interrupt to an EL2 destination context.
- Such instructions are not being present in the source context.
- IMP_CPUACTLR_EL1.DMB bit is set which enables such instructions to be interruptible.
- The memory translation regime at the target Exception level uses PMSAv8-64.
- There are no ongoing accesses to debug or trace functions, *Reliability, Availability, and Serviceability* (RAS) registers, *Performance Monitoring Unit* (PMU), or timer registers.
- The targeted core is not in the process of correcting an *Error Correcting Code* (ECC) error.
- The cluster and the targeted core are powered up and that the core clock SCLK is running. In other words, the core is not executing a *WFI* or *WFE* instruction.
- Exception-handling software is not using implicit error synchronization events, that is SCTLRL_ELx.IESB is assumed to be 0b0.
- No core in the cluster is sending *Software Generated Interrupts* (SGI).
- No external agent (such as a DMA engine) is accessing the targeted core's *Tightly Coupled Memories* (TCMs) through the *ACE-Lite Subordinate* (ACELS) port.
- No external agent (such as a DMA engine) is accessing the targeted core's *Low-latency RAM* (LLRAM) interface through the ACELS port.
- The targeted core is not in the process of handling another interrupt.
- No core in the cluster is performing any Device memory accesses.
- No core in the cluster is performing any atomic operations.
- No core in the cluster is performing any exclusive operations.

8.14.1.3 Worst-case interrupt latency conditions

Worst-case interrupt latency conditions allow for the interrupt recognition to be delayed.

This may occur because:

- The core is executing a Device, atomic or exclusive operation.
- The core currently handling an interrupt and therefore masking new interrupts.
- Interrupts are being delivered to other cores in the cluster.

The Cortex®-R82 processor accepts and processes the interrupt within 120 SCLK cycles if your system and software meet the following conditions:

- The GIC distributor used is GIC-625.
- The GIC distributor is clocked at half the frequency as the Cortex®-R82 cluster, with no bus bridges between the two.
- PERIPHCLK is clocked at 25% or higher of the SCLK frequency.
- The *Shared Peripheral Interrupt* (SPI) which is asserted is a low-latency targeted SPI.

- Interrupt configuration (routing, priority, group assignment, per-interrupt enables, group enables, exception vector) is static.
- The level interrupts are not deasserted before they are serviced.
- Data memory barrier instructions and instructions that write to memory with release semantics do not delay interrupts. That is, either of the following applies:
 - An EL0 or EL1 source context takes an interrupt to an EL2 destination context.
 - Such instructions are not being present in the source context.
 - IMP_CPUACTLR_EL1.DMB bit is set which enables such instructions to be interruptible.
- The memory translation regime at the target Exception level uses PMSAv8-64.
- There are no ongoing accesses to debug or trace functions, RAS registers, or PMU registers.
- The targeted core is not in the process of correcting an *Error Correcting Code* (ECC) error.
- Any core in the cluster can be executing a `WFI` or `WFE` instruction, in which case they may have their clock being architecturally gated. However, the Cortex®-R82 processor SCLK must be active. This can be achieved by setting IMP_CLUSTERACTLR_EL1.SCLKQ to 0b1.
- Exception-handling software is not using implicit error synchronization events, that is SCTLX_ELx.IESB is assumed to be 0b0.
- Cores in the cluster are allowed to send *Software Generated Interrupts* (SGI).
- External agents (such as a DMA engine) are allowed to be accessing any core's TCM memories through the *ACE-Lite Subordinate* (ACELS) port.
- External agents (such as a DMA engine) are allowed to be accessing any core's *Low-latency RAM* (LLRAM) interface through the ACELS port.
- Cores are allowed to perform Device memory accesses. Device memory accesses are only performed through the *Low-latency Peripheral Port* (LLPP) or the *Shared Peripheral Port* (SPP) and not through the LLRAM or *Main Manager* (MM) ports. Device memory accesses do not span the 128-bit boundary for the LLPP and 64-bit boundary for the SPP.
- Cores are allowed to perform Atomic operations. Atomic operations may be performed through the LLRAM or MM ports but they must be configured to perform them as near atomics. This is done by setting the IMP_CPUACTLR_EL1.ATOM bit field to 0b01.
- Cores are allowed to perform Exclusive operations. Exclusive operations may be performed through the LLRAM or MM ports, but they must be configured to be cacheable within the cluster (The *Memory Protection Unit* (MPU) regions are programmed as cacheable and Inner Shareable).
- Atomic and exclusive operations are allowed to be performed to the *Tightly Coupled Memories* (TCMs). Such operations cannot be used for inter-core communication because the TCM regions are private to each core.
- The system components attached to the Cortex®-R82 processor manager ports respond in a timely fashion. That is, the components are clocked synchronously to SCLK with a 2:1 clock ratio; the components can handle pipelined transactions; and the components respond within 3 clock cycles.
- There can be any number of cores in the cluster (up to the maximum of eight cores). Interrupts can be under delivery to any number of cores in the cluster.

- SPIs can be asserted at the same time which target any number of cores within the cluster.
- Ongoing accesses to timer registers are allowed.
- The new interrupt, for which the latency is measured, has higher priority than any other ongoing interrupt that is being processed.

The worst-case interrupt latency conditions assume that the software running on the Cortex®-R82 processor fulfills the conditions set above for getting 120 SCLK cycle interrupt response. To ensure these conditions are met, the hypervisor running at EL2 can enforce the EL1 and EL0 software to achieve some of those conditions.

- Setting the IMP_INTLATENCY_EL2.DEV bit to 0b1 forces any Device access on the LLRAM or MM ports to abort.
- Setting the IMP_INTLATENCY_EL2.DEV bit to 0b1 forces any Device memory access where all bytes are not within a 128-bit aligned region on the LLPP or a 64-bit aligned region on the SPP to abort.
- Setting the IMP_INTLATENCY_EL2.ATOM bit to 0b1 forces atomics either to be near or to abort if they cannot be executed near.
- Setting the IMP_INTLATENCY_EL2.EXCL bit to 0b1 forces exclusives that cannot be contained in the cluster to abort.
- The EL2 MPU region programming can also override EL1 MPU region programming.

See the [A.2.2.106 IMP_INTLATENCY_EL2, Interrupt Latency Register](#) on page 707 for more information.

8.14.2 Real-time hierarchy

The Cortex®-R82 processor provides various memories and interfaces each tailored to different real-time requirements. The aim is that some memories and interfaces are used for more critical real-time requirements and some for less critical real-time requirements.



Note

The more real-time critical context is also able to access the less real-time critical interfaces and memories although such an access might not be desirable depending on the system design.

The Cortex®-R82 processor memories and interfaces can be ordered as follows in terms of meeting the critical real-time requirements:

1. The *Tightly Coupled Memories* (TCMs) and the *Low-latency Peripheral Port* (LLPP) are suitable for meeting the most critical real-time requirements.
2. The *Low-latency RAM* (LLRAM) port, cached through the L1 caches, and the *Shared Peripheral Port* (SPP) are suitable for meeting the medium critical real-time requirements. They are more deterministic than the *Main Manager* (MM) but less deterministic than the TCMs and the LLPP.
3. MM, cached through the L1 and L2 caches, is suitable for meeting the least critical real-time requirements. MM is the least deterministic.

To ensure that the LLRAM and the SPP accesses meet higher real-time requirements compared to the MM accesses, the Cortex®-R82 processor provides a mechanism to reserve buffers for the LLRAM and SPP accesses. Reserving certain buffers specifically for LLRAM and SPP accesses avoids the cases where all the buffers are in use for the MM accesses while the software tries to access the LLRAM or the SPP, therefore avoids the need to wait for ongoing MM accesses to complete.

If the Cortex®-R82 processor has the LLRAM port and the SPP implemented and if your system always requires the LLRAM and SPP accesses to have higher real-time response than the MM accesses:

- The EL1 software should set the `IMP_CPUACTLR_EL1.LCURES` to `0b1` to reserve buffer slots and linefill descriptors to ensure that the LLRAM and SPP accesses do not have increased worst case latency caused by MM accesses.
- Hypervisor software running at EL2 can also set the `IMP_INTLATENCY_EL2.LCURES` to `0b1` to ensure that the EL1 software treats the `IMP_CPUACTLR_EL1.LCURES` as `0b1` regardless of its actual value.

The default values for `IMP_CPUACTLR_EL1.LCURES` and `IMP_INTLATENCY_EL2.LCURES` bits are `0b0`. If your system requires the LLRAM and SPP accesses to be always prioritized over MM accesses, consider setting some of these bits to `0b1`, according to your needs.

See [A.2.2.44 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register](#) on page 512 and [A.2.2.106 IMP_INTLATENCY_EL2, Interrupt Latency Register](#) on page 707 for more information.

8.14.3 Freedom from Interference

This section details how the Cortex®-R82 processor can be used to create systems which achieve freedom from interference.

8.14.3.1 Temporal Freedom from Interference

A system achieves temporal freedom of interference when it enables each system task to meet response time deadlines by providing predictable and timely access to critical system resources, such as computation, memory and peripherals.

The Cortex®-R82 processor will aim to ensure freedom of interference between different contexts where possible, where freedom of interference is defined as the ability for a more deterministic context to make forward progress without dependencies on less deterministic contexts. In particular a real-time context should be free of any interference from a context running a rich guest OS. When multiple real time contexts exist within a system it is also possible for there to be dependencies between them depending on the system design.

The Cortex®-R82 processor supports multiple memory ports which provide differing levels of determinism, for the purpose of describing how to achieve temporal freedom from interference the following three usage models are described in this section:

1. Non real-time context, which uses the main manager (MM) port for Normal memory accesses, and the shared peripheral port (SPP) for Device memory accesses. A non real-time context can use VMSA or PMSA.
2. Partially real-time context, which uses the low-latency RAM (LLRAM) port for shared Normal memory accesses, tightly-coupled memories (TCM) for non-shared Normal memory accesses, SPP for access to shared peripherals with Device memory accesses and the low-latency peripheral port (LLPP) for access to peripherals private to the core. A partially real-time context can only use PMSA.
3. Fully real-time context, which only uses TCM for Normal memory accesses and LLPP for Device memory accesses. This provides the most deterministic execution. A fully real-time context can only use PMSA.

A context refers to either software running at EL1 and EL0 with a given value of VSCTLR_EL2.VMID set or software running at EL2. All software running at EL2 is within the same context. If a system uses multiple contexts, switching between them is expected to be controlled by a hypervisor at EL2. For a system that does not utilize EL2 it is expected that only a single context is executed and so freedom from interference between contexts does not apply.

When developing software or systems using the Cortex®-R82 processor, it should be noted:

- The architecture only guarantees that a single PE in the system makes forward progress with an exclusive sequence. Exclusives are unlikely to be suitable when it is required that all exclusive threads make forward progress. This is due to the architecture of exclusive sequences and is not specific to the Cortex®-R82 processor. It is recommended to use atomics instead to implement synchronization primitives such as semaphores.
- When software meets the assumptions of use for freedom from interference, it will ensure that a more deterministic context does not have a dependency on a less deterministic context. However, a less deterministic context may still affect the performance of a more deterministic context.
- By dedicating a context to a particular core, performance may be improved as more resources are available to that context. This can also allow improved fault containment, as if only a single context is running on a core any asynchronous faults that occur can be contained to that context.
- As freedom from interference is a superset of the requirements to guarantee interrupt latency, not all types of instruction can be used. For example, non-cacheable atomics cannot be interrupted and so it is not possible to guarantee freedom from interference when they are used. This means that not all types of instruction are available for sharing data while guaranteeing freedom from interference.

8.14.3.2 Temporal freedom from interference between a rich context and a partially real-time context

This section details how the Cortex®-R82 processor can be used to create systems which achieve temporal freedom from interference between a rich context and a partially real-time context.

Freedom from interference between a rich context from partially real-time context will be achieved if your system and software meets the following conditions:

- A partially real-time context does not use the MM port.
 - The MM port is not designed to provide freedom from interference from other cores and from dependencies in the system.
 - Any VMSA memory access will have an implicit dependency on the MM port, so a partially real-time context must only use PMSA.
 - This includes not executing DC and IC instructions associated with MM addresses.
 - This includes not executing IC IALL{US}, or any TLBI instructions.
 - Any context that uses MM and executes a DSB SY will depend on previous instruction cache and TLB maintenance performed in any context.
 - Enforcement: This can be enforced using the stage 2 MPU, IC IALL{US} instructions can be prevented by setting IMP_INTLATENCY_EL2.MMDVM and TLBI instructions can be prevented by setting HCR_EL2.TTLB.
- Memory is not shared between contexts, unless following all the assumptions for how to share data between contexts.
 - Enforcement: This can be enforced using the stage 2 MPU and setting IMP_INTLATENCY_EL2.MMDVM.
- The LLRAM port is connected to a simple SRAM, or the system otherwise ensures LLRAM transactions are always able to make forward progress without depending on other less deterministic transactions in the system.
 - Enforcement: This must be enforced by the system designer.
- Peripherals connected to the SPP and LLPP shall not share resources with other buses in the system or otherwise ensure that SPP and LLPP transactions are always able to make forward progress without depending on other less deterministic transactions in the system.
 - Enforcement: This must be enforced by the system designer.
- The IMP_CPUACTLR_EL1.DMB or IMP_INTLATENCY_EL2.DMB bit is set to upgrade Data Memory Barrier (DMB) instructions into a Data Synchronization Barrier (DSB) allowing it to be flushed on a context switch and so not delay execution of barriers in the new context.
 - Enforcement: This can be enforced by IMP_INTLATENCY_EL2.DMB
- Set/way maintenance of the L1 data cache is not used as this introduces a dependency on the MM port.
 - Enforcement: This can be enforced by HCR_EL2.TSW.
- The IMP_CPUACTLR_EL1.LCURES or IMP_INTLATENCY_EL2.LCURES bit is set.
 - Enforcement: This can be enforced by setting IMP_INTLATENCY_EL2.LCURES
- Software follows the assumptions of use for worst-case interrupt latency conditions.
- There are no ECC errors
- There are no bus timeouts on LLRAM
 - It is expected that recovery software can run using TCM and LLPP after a bus timeout, but it is not expected that real-time contexts will be able to continue to run as it is likely not to be possible to perform a context switch after a bus timeout has occurred.

- When a bus timeout occurs, it is not expected that the old context will be saved and so it is not possible to return to the context in which the bus timeout occurred and further accesses to LLRAM will abort.
- Assumes that all cores in the cluster follow assumptions on freedom from interference

8.14.3.3 Temporal freedom of interference between a fully real-time context and less deterministic contexts

This section details how the Cortex®-R82 processor can be used to create systems which achieve temporal freedom from interference.

Freedom from interference between a fully real-time context and less deterministic contexts will be achieved if your system and software meets the following conditions:

- A fully real-time context only uses the TCM and LLPP ports.
 - Enforcement: This can be enforced using the stage 2 MPU, or by disabling the regions in the region registers and removing write access to the region registers.
- Memory is not shared between contexts, unless following all the assumptions for how to share data between contexts.
 - Enforcement: This can be enforced using the stage 2 MPU.
- Peripherals connected to the LLPP shall not share resources with other buses in the system or otherwise ensure that LLPP transactions are always able to make forward progress without depending on other less deterministic transactions in the system.
 - Enforcement: This must be enforced by the system designer.
- The IMP_CPUACTLR_EL1.DMB or IMP_INTLATENCY_EL2.DMB bit is set to upgrade Data Memory Barrier (DMB) instructions into a Data Synchronization Barrier (DSB) allowing it to be flushed on a context switch and so not delay execution of barriers in the new context.
 - Enforcement: This can be enforced by IMP_INTLATENCY_EL2.DMB.
- Maintenance of caches or TLBs are not performed.
 - Enforcement: This can be enforced by setting all of HCR_EL2.{TTLB, TPU, TPCP, TSW}.
- Software follows the assumptions of use for worst-case interrupt latency conditions.
- There are no bus timeouts on LLPP.
 - It is expected that recovery software can run using TCM and LLPP after a bus timeout, but it is not expected that real-time contexts will be able to continue to run as it is likely not to be possible to perform a context switch after a bus timeout has occurred.
 - When a bus timeout occurs, it is not expected that the old context will be saved and so it is not possible to return to the context in which the bus timeout occurred and further accesses to LLPP will abort.
- Assumes that all cores in the cluster follow assumptions on freedom from interference.

8.14.3.4 Ensuring temporal freedom of interference in a hypervisor

This section details how the Cortex®-R82 processor can be used to create systems which achieve temporal freedom from interference when using a hypervisor to manage contexts.

Freedom from interference when a hypervisor is present will be achieved if your system and software meets the following conditions:

- The hypervisor follows all assumptions to ensure freedom from interference according to the memory ports used by the hypervisor.
 - When the hypervisor is only using TCM and LLPP, assumes the hypervisor follows all assumptions to ensure freedom from interference from a fully real-time context from other contexts.
 - When the hypervisor is also using LLRAM and/or SPP, assumes the hypervisor follows all assumptions to ensure freedom from interference from a partially real-time context from other contexts.
 - Assumes that the hypervisor does not use the MM port. It is possible to use the MM port to boot from, then branch to execution from another port and execute a DFB instruction. When the DFB instruction is completed it is then possible to achieve freedom from interference.
- The Data Full Barrier (DFB) instruction is not used unless the software after the DFB does not rely on freedom from interference.
 - DFB will depend on all previous memory accesses but is interruptible, so only creates a dependence for the instructions after it in program order and if there is an interrupt the DFB will be flushed and so not create any dependencies. Once the DFB instruction is completed it has no impact on freedom from interference.
- AT operations for EL1/0 are not executed in EL2 when the VTCR_EL2.MSA bit is set.
 - This will introduce dependencies on MM by executing a pagewalk.
- The hypervisor does not execute cache debug operations and ensures ACTLR_EL2.CDBG is cleared.
- The hypervisor only switches between two contexts on a given core.
- The hypervisor sets the IMP_INTLATENCY_EL2.LCUDVM bit when non real-time contexts are executing to prevent introducing dependencies when data is shared between different contexts.
- The hypervisor sets the IMP_INTLATENCY_EL2.MMDVM bit when real-time contexts are executing to prevent introducing dependencies when data is shared between different contexts.
- The hypervisor disables bus timeout aborting behavior for ports that are not in use when switching to a new context.
 - For example, when switching to a real-time context the hypervisor should disable aborting behavior for bus timeouts on MM. Any bus timeouts for not currently active contexts will be reported in the RAS registers.

8.14.3.5 Ensuring temporal freedom of interference while sharing data

This section details how the Cortex®-R82 processor can be used to create systems which achieve temporal freedom from interference while sharing data.

Freedom from interference while sharing data between different contexts will be achieved if your system and software meets the following conditions:

- Data is only shared between the following pairs of contexts:
 - Between a non-real time and a partially real-time context using the LLRAM port.
 - Between a partially real-time and a fully real-time context using TCMs.
- Instruction cache maintenance is not performed to the LLRAM port when sharing data using the LLRAM port on the same core.
 - Enforcement: This can be enforced by setting IMP_INTLATENCY_EL2.LLRAMDVM.

Freedom from interference while sharing data between different real-time and non real-time contexts can be performed without introducing dependencies by using one of the following methods:

- The non real-time context executes writes into the LLRAM memory by performing writes to the MM port targeting the ACELS-LLRAM via a loopback in the system.
 - Note: this requires functionality provided by the system.
 - Enforcement: This can be enforced by the stage 2 MPU.
- Contexts are statically allocated to cores such that all contexts on a given core have the same real-time requirements.
 - For example, one core can be executing a rich OS which also produces some data, written to LLRAM, to be consumed by a partially real-time context. The consumer is allocated to a different core which reads from this same location from the LLRAM port.
 - Enforcement: This can be enforced by the hypervisor.
- The non real-time context uses LLRAM to read or write data, without performing instruction cache maintenance for LLRAM addresses.
 - Enforcement: This can be enforced by setting IMP_INTLATENCY_EL2.LLRAMDVM.

8.14.4 Quality of Service

You can prioritize memory accesses from a specific core over accesses from the other cores within the Cortex®-R82 processor. Similarly, you can also prioritize the Cortex®-R82 cluster over the other clusters within your system.

You can set priority options for a core or for the whole cluster with the IMP_CLUSTERQOSR_EL1 register.

Setting the IMP_CLUSTERQOSR_EL1.COREQOSEN to 0b1 enables *Quality of Service* (QOS) within the Cortex®-R82 processor. When this bit is set to 0b1, you can set which core is prioritized by

setting the IMP_CLUSTERQOSR_EL1.COREQOSID bits. The COREQOSID bitfield has no effect if the COREQOSEN is 0b0.

When QOS is enabled for a specific core, the *Low-latency RAM* (LLRAM), the *Shared Peripheral Port* (SPP), and the L2 cache accesses from that core have higher priority over the accesses from the other cores within the Cortex®-R82 processor.

IMP_CLUSTERQOSR_EL1.CLQOS controls the priority over other clusters within your system. This field is driven by the *Main Manager* (MM) ARQOSM and AWQOSM QOS signals. Your interconnect should support prioritization of clusters if your system requires the QOS to be set for the whole cluster.

See [A.2.2.57 IMP_CLUSTERQOSR_EL1, Cluster Quality of Service Register](#) on page 566 for more information.

9. Memory management

This chapter describes how the memory space of the Cortex®-R82 processor is managed.

9.1 About the memory management

The Cortex®-R82 processor memory management system determines various attributes for each memory location including access permissions, memory types, and Cacheability.

Access permissions indicate which levels of privilege are permitted to access a location and whether write access or instruction execution are permitted. Memory type and Cacheability attributes affect the way the Cortex®-R82 processor handles particular accesses, for example, whether or not it permits two stores to be merged into a single write access.

Each core within the Cortex®-R82 processor has two programmable *Memory Protection Units* (MPUs) and an optional *Memory Management Unit* (MMU):

- The EL1 MPU is controlled by operating system software running at EL1. The EL1 MPU enables isolation between applications running at EL0.
- The EL2 MPU is controlled by hypervisor software running at EL2. The EL2 MPU enables isolation between operating systems running at EL1.
- Optional EL1 MMU is controlled by operating system software running at EL1. The EL1 MMU enables address translation and isolation between applications running at EL0.

The MPU implements the *Protected Memory System Architecture* (PMSA) and MMU implements the *Virtual Memory System Architecture* (VMSA).

PMSA has no address translation capabilities. Therefore the physical address is always the same as the virtual address for the MPU.

VMSA has address translation capabilities and is responsible for translating virtual addresses into physical addresses.



Virtual address refers to the address before the translation process as generated by the instruction. Physical address refers to the address after the translation process as visible on the bus.

EL2 software goes through one stage of EL2 MPU translation. EL1 and EL0 software go through one stage of either EL1 MPU or EL1 MMU translation optionally followed by one stage of EL2 MPU translation.



- If the EL1 MMU is not included, then the EL1 MPU has non zero regions. If the EL1 MMU is included, then the EL1 MPU is optional.
- EL2 MPU programmable regions are optional. The value 0 for MPU region indicates that the core does not support any programmable regions for this MPU. Note however that default or architectural memory maps may still be used.

The Cortex®-R82 processor memory management architecture enables virtualization of operating systems. Hypervisor software running on EL2 selects between the MPU and MMU on a per-operating system basis.

The Cortex®-R82 processor always operates in Secure state and all translation regimes are Secure. However, the Cortex®-R82 processor is able to access both Secure and Non-secure address space. The Cortex®-R82 processor MPU regions and MMU pages include configurable attributes that determine whether accesses to addresses within a region or page are to the Secure or Non-secure address space.

Memory management system translation results are cached to reduce translation costs on performance and power. Each core within the Cortex®-R82 processor includes an L1 instruction cache structure (L1I MMS) and an L1 data cache structure (L1D MMS) that contain the results of memory management system lookups. An access that hits in the L1I MMS or L1D MMS incurs no extra latency.

Each core within the Cortex®-R82 processor that supports VMSA includes a L2 *Translation Lookaside Buffer* (TLB) that contains the results of page table walks.

9.2 MPU

Each core within the Cortex®-R82 processor has two programmable *Memory Protection Units* (MPUs), controlled from EL1 and EL2. Depending on the parameter PA_W, each MPU supports a 40-bit (when PA_W=40) or a 48-bit (PA_W=48) physical address range that allows up to 1TB and 256TB memory address range, respectively, to be subdivided into regions.

Each memory region is defined by a base address, limit address, access permissions, and memory attributes.

For data accesses, the MPU checks that the type of access (read or write) to a region is allowed for the current translation regime. For instruction accesses, the MPU checks if an access is allowed to the region and that the translation regime allows execution. For both data and instruction accesses, if access is allowed, the MPU assigns the memory attributes defined for the region. If access is not allowed, a permission fault is taken. A translation fault is taken for the following reasons:

- If an access hits in more than one region in one of the MPUs.
- If an access does not hit in any MPU region and the Background region cannot be used (based on the MPU configuration and current privilege level).



A translation fault is only taken when the MPU is enabled in software.

As a result of pipelined operation, the Cortex®-R82 processor tries to predict program flow and future data accesses, and so it fetches data and instructions ahead of their use. These transactions are known as speculative transactions until the pipeline completes execution of the corresponding instruction. This might result in the Cortex®-R82 processor generating addresses either outside permitted regions or not having privilege to attempt the access. In these cases, speculative accesses are prevented from generating bus transactions by the MPU but do not raise a translation or permission fault.

Each core within the Cortex®-R82 processor has an EL1-controlled MPU and an EL2-controlled MPU with 0, 16, or 32 programmable regions. The value 0 indicates that the core does not include support for MPU for any programmable regions, but software can still make use of architectural/default memory maps. For the EL1-controlled MPU, the value 0 is supported only when the core includes support for *Memory Management Unit* (MMU).

When the EL2-controlled MPU and virtualization are enabled, all transactions using the EL0/EL1 translation regime perform a lookup in both MPUs. The resulting attributes are combined so that the least permissive attributes are taken. These two stages of protection allow the hypervisor to retain control over the EL0/EL1 translation regime and therefore enables support for virtualization. When software executes using the EL2 translation regime, only the EL2-controlled MPU is used.

For more information on the MPU, see the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

9.2.1 MPU regions

A region is a contiguous range of addresses starting at a base address, extending up to and including a limit address.

The Cortex®-R82 processor *Memory Protection Unit* (MPU) region address ranges are defined as a pair with base and limit addresses. These addresses are arbitrary except for a minimum 64-byte resolution. This provides flexibility and reduces the number of regions that are required to fully describe a memory map compared to older versions of the Arm architecture.

The base address is configured by PRBAR_EL1 (PRBAR_EL2 for EL2-controlled MPU) and the limit address is configured by PRLAR_EL1 (PRLAR_EL2 for EL2-controlled MPU). The base address is aligned on a 64-byte boundary and the limit address is aligned to the byte below a 64-byte boundary. For the remainder of this section, short terms such as PRBAR are used to describe any of the EL1 or EL2 MPU registers (PRBAR_EL1, PRBAR_EL2, PRBAR<n>_EL1, PRBAR<n>_EL2).

Both base and limit addresses are inclusive, meaning that an address within a region is given by:

```
PRBAR.BASE:0b000000 <= address <= PRLAR.LIMIT:0b111111
```

Where : is a bit concatenation operator.

The minimum size for a region is 64 bytes.

The Cortex®-R82 processor implements a parameterized physical address width, and can be either 40 (when parameter PA_W=40) or 48 (when parameter PA_W=48).



Even though the minimum resolution that can be programmed in the MPU region is 64 bytes, the Cortex®-R82 processor is optimized for a minimum region size of 4KB. Any region smaller than 4KB may incur additional cycles of latency for instruction fetches, loads, and stores.

The Cortex®-R82 processor MPU regions support programming through System register operations. Both an indirect method (for example through PRSEL_{EL1}/PRBAR_{EL1}/PRLAR_{EL1}) and a direct method (for example through PRBAR_{<n>_EL1}/PRLAR_{<n>_EL1}) are provided. This allows software to be optimized for either code density or faster reprogramming and context switching.

PRBAR and PRLAR also hold the access permissions (PRBAR.AP), Shareability (PRBAR.SH), the Execute-never bit (PRBAR.XN), and memory attribute index (PRLAR.AttrIdx).

Memory attributes are determined by indexing the *Memory Attribute Indirection Registers* (MAIR_{ELx}) with PRLAR.AttrIdx.

A region is enabled or disabled by setting or clearing the region enable bit (PRLAR.EN). In the EL1-controlled and EL2-controlled MPUs, regions can also be enabled or disabled by writing to the *MPU Region Enable Register* (PRENR_{EL1} or PRENR_{EL2}).

Speculative memory accesses can occur as a result of prefetching instructions or predicting data accesses. The Cortex®-R82 processor only speculates on Normal memory, and does not speculate on Device memory. If your system needs to avoid speculative accesses for certain address regions, ensure that the combined effect from MPU programming, Background region, and the related system control bits result in these address regions either producing a fault or assigning a Device memory attribute.

9.2.1.1 EL1-controlled MPU Background region

When the EL1-controlled *Memory Protection Unit* (MPU) is disabled (SCTLR_{EL1}.M=0):

- If SCTLR_{EL1}.BR =1, the EL1 default memory map (MPU Background region) is used as it is shown in [Table 9-1: EL1-controlled MPU Background region - instruction access and data access](#) on page 199.
- If SCTLR_{EL1}.BR =0, the Arm®v8-A AArch64 Memory View is used.

When the EL1-controlled MPU is enabled (SCTLR_EL1.M=1), the MPU Background region can be enabled by setting SCTLR_EL1.BR. In this case, accesses from the EL1 translation regime that do not hit any programmable regions use the EL1-controlled MPU Background region.

The complete EL1-controlled MPU Background region is always Secure, that is, Non-secure (NS) bit is 0.

The following table shows the EL1-controlled MPU Background region for both instruction access and data access.

Table 9-1: EL1-controlled MPU Background region - instruction access and data access

Region	Address range	Attributes	Execute-never (XN) bit
0	0x00000000–0x3FFFFFFF	Normal, Inner Write-Back, Inner Read-Allocate, Inner Write-Allocate, Outer Write-Back, Outer Read-Allocate, Outer Write-Allocate, Outer Shareable	0
1	0x40000000–0x7FFFFFFF	Normal, Inner Write-Through, Inner Read-Allocate, Inner Write-Allocate, Outer Write-Through, Outer Read-Allocate, Outer Write-Allocate, Outer Shareable	0
2	0x80000000–0xFFFFFFFF	Normal, Inner Non-cacheable, Outer Non-cacheable, Outer Shareable	1
3	0x000100000000–0x00FFFFFFFF	Device-nGnRnE	1

Access permission for all regions and address ranges is 0b00. Access permission 0b00 implies that the EL1 MPU Background region provides both read and write access to accesses from EL1 translation regime that hit any region in the EL1 MPU Background region.

Any accesses from EL0 that hit in EL1 MPU Background region have no read/write access and generate permission fault.

9.2.1.2 EL2-controlled MPU Background region

When the EL2-controlled *Memory Protection Unit* (MPU) is disabled (SCTLR_EL2.M=0):

- If SCTLR_EL2.BR =1, the EL2 default memory map (MPU Background region) is used as it is shown in [Table 9-2: EL2-controlled MPU Background region - instruction access and data access](#) on page 200.
- If SCTLR_EL2.BR =0, the Arm®v8-A AArch64 Memory View is used.

When the MPU is enabled (SCTLR_EL2.M=1), the MPU Background region can be enabled by setting SCTLR_EL2.BR. In this case, accesses from the EL2 translation regime that do not hit any programmable regions use the EL2-controlled MPU Background region.

The complete EL2-controlled MPU Background region is always Secure, that is, Non-secure (NS) bit is 0.

The following table shows the EL2-controlled MPU Background region for both instruction access and data access.

Table 9-2: EL2-controlled MPU Background region - instruction access and data access

Region	Address range	Attributes	Execute-never (XN) bit
0	0x00000000–0x3FFFFFFF	Normal, Inner Write-Back, Inner Read-Allocate, Inner Write-Allocate, Outer Write-Back, Outer Read-Allocate, Outer Write-Allocate, Outer Shareable	0

Region	Address range	Attributes	Execute-never (XN) bit
1	0x40000000–0x7FFFFFFF	Normal, Inner Write-Through, Inner Read-Allocate, Inner Write-Allocate, Outer Write-Through, Outer Read-Allocate, Outer Write-Allocate, Outer Shareable	0
2	0x80000000–0xFFFFFFFF	Normal, Inner Non-cacheable, Outer Non-cacheable, Outer Shareable	1
3	0x000100000000–0x00FFFFFFFF	Device-nGnRnE	1

Access permission for all regions and address ranges is 0b00. Access permission 0b00 implies that the EL2 MPU Background region provides both read and write access to accesses from EL2 translation regime that hit any region in the EL2 MPU Background region.

Any accesses from EL1 that hit in EL2 MPU Background region have no read/write access and generate permission fault.

9.2.1.3 Default Cacheability

When default Cacheability is enabled (HCR_EL2.DC=1), transactions using the EL1-controlled MPU Background region have Normal, Inner Write-Back, Outer Write-Back, Non-shareable attributes applied with both Read-Allocate and Write-Allocate hints enabled. Instruction accesses that hit in the Background region when HCR_EL2.DC=1 are always executable.

The default attributes are the most permissive, meaning that when combined with any attribute from the EL2-controlled MPU the resulting attribute is the same as the EL2-controlled MPU attribute. This allows the EL2-controlled MPU to effectively make the EL1-controlled MPU transparent to transactions from the EL1 translation regime that hit in the Background region. When HCR_EL2.DC=1, all translations from the EL0/EL1 translation regime perform a two-stage MPU lookup and the Cortex®-R82 processor behaves as if HCR_EL2.VM is set.

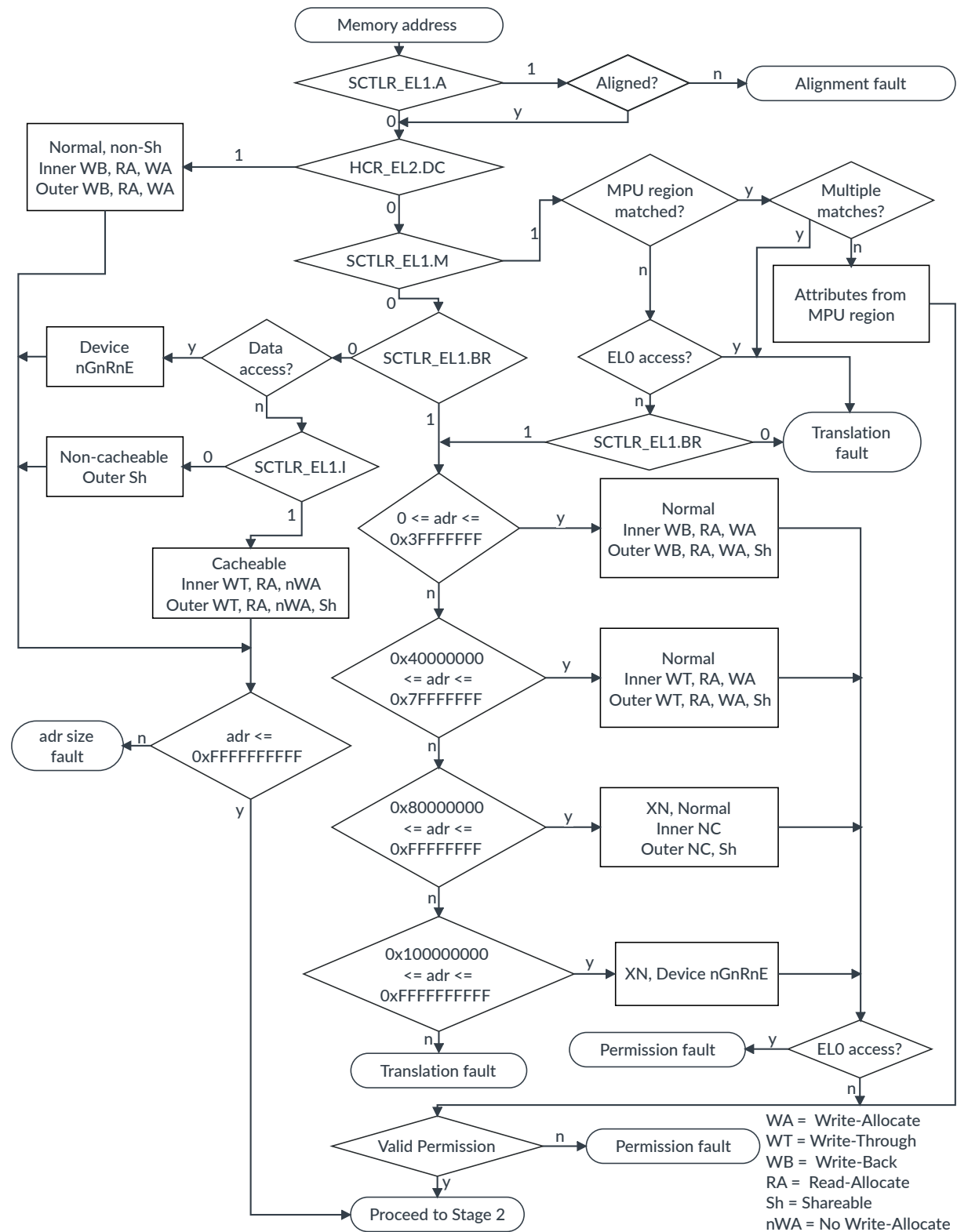
9.2.1.4 Combined MPU Checking flowchart

The following figure shows EL1 *Memory Protection Unit* (MPU) checking flowchart.



Note

The figure is not exhaustive. Other System register fields might influence the Effective value of the ones mentioned on the figure, or some instructions might behave differently. For more information on the detailed description of the System register fields, see [A. AArch64 registers](#) on page 279.

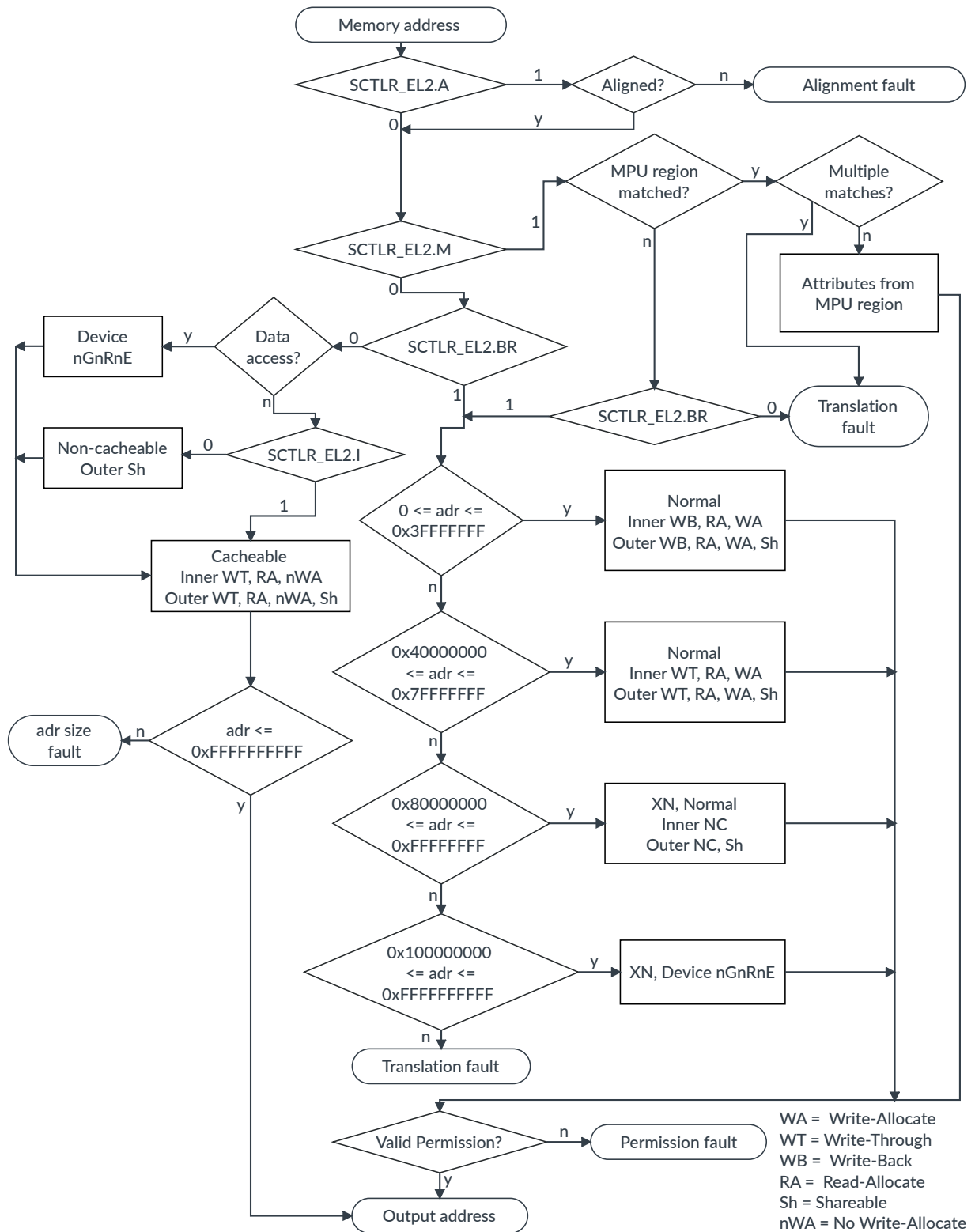
Figure 9-1: EL1 MPU check

The following figure shows EL2 MPU checking flowchart.



Note

The figure is not exhaustive. Other System register fields might influence the Effective value of the ones mentioned on the figure, or some instructions might behave differently. For more information on the detailed description of the System register fields, see [A. AArch64 registers](#) on page 279.

Figure 9-2: EL2 MPU check

9.2.2 Virtualization support

To support virtualization, two stages of MPU lookup are performed.

Virtualization allows processes running at EL1 and EL0 (typically one or more guest operating systems and their applications) to be managed by processes running at EL2 (typically a single hypervisor).

The EL1-controlled MPU checks transactions from processes running at EL0 or EL1 and is programmed by processes running at EL1 or EL2. The EL2-controlled MPU also checks transactions executed from the EL0/EL1 translation regime when virtualization is enabled and programmed by software at EL2. Transactions executed under the EL2 translation regime use the EL2-controlled MPU only.

When virtualization is enabled (HCR_EL2.VM=1) and the EL2-controlled MPU is enabled (SCTLR_EL2.M=1), transactions permitted by the EL1-controlled MPU are checked by the EL2-controlled MPU as part of a two stage lookup. If both MPUs permit the transaction, memory attributes from stage 1 are combined with attributes from the matching region in stage 2 and the stricter of the two sets of attributes are applied to the transaction.

9.2.2.1 Combining MPU memory attributes

When a two-stage lookup is performed, the memory type, Cacheability, and Shareability attributes from each MPU are combined.

Combining the memory type attribute

The following table shows how the memory type assignments are combined under typical conditions as part of a two-stage lookup.

Table 9-3: Combining the memory type assignments

Assignment in EL1-controlled MPU	Assignment in EL2-controlled MPU	Resultant type
Device-nGnRnE	Any	Device-nGnRnE
Device-nGnRE	Device-nGnRnE	Device-nGnRnE
	Not Device-nGnRnE	Device-nGnRE
Device-nGRE	Device-nGnRnE	Device-nGnRnE
	Device-nGnRE	Device-nGnRE
	Not (Device-nGnRnE or Device-nGnRE)	Device-nGRE
Device-GRE	Device-nGnRnE	Device-nGnRnE
	Device-nGnRE	Device-nGnRE
	Device-nGRE	Device-nGRE
	Device-GRE or Normal	Device-GRE
Normal	Any type of Device	Device type assigned at stage 2
	Normal	Normal

Force Write-Back

The attributes from the stage 1 and stage 2 translation regimes are typically combined to be the most restrictive of the two. For example, if the stage 1 translation regime returns a Device attribute and the stage 2 translation regime returns Normal memory attributes, the final combined attributes become the stricter of the two, which is Device.

However, if HCR_EL2.FWB bit is set, the Hypervisor forces the final attributes to be Normal, Inner and Outer Write-Back when the EL2 translation regime returns Normal, Inner and Outer Write-Back. This implies that the stage 1 translation regime attributes are ignored even if they were marked as Device. This is particularly useful when the Hypervisor wants to limit the interrupt latency window by relaxing the Device attributes.



Forcing Device attribute to Normal memory allows the Cortex®-R82 processor to make speculative accesses to peripherals, which may be undesirable.

Combining the Cacheability attribute

The following table shows how the Cacheability assignments are combined as part of a two-stage lookup.

Table 9-4: Combining the Cacheability assignments

Assignment in EL1-controlled MPU	Assignment in EL2-controlled MPU	Resultant Cacheability
Non-cacheable	Any	Non-cacheable
Any	Non-cacheable	Non-cacheable
Write-Through Cacheable	Write-Through or Write-Back Cacheable	Write-Through Cacheable
Write-Through or Write-Back Cacheable	Write-Through Cacheable	Write-Through Cacheable
Write-Back Cacheable	Write-Back Cacheable	Write-Back Cacheable

Combining the Shareability attribute

The following table shows how the Shareability assignments are combined as part of a two-stage lookup.

Table 9-5: Combining the Shareability assignments

Assignment in EL1-controlled MPU	Assignment in EL2-controlled MPU	Resultant Cacheability
Outer Shareable	Any	Outer Shareable
Inner Shareable	Outer Shareable	Outer Shareable
	Inner Shareable	Inner Shareable
	Non-shareable	Inner Shareable
Non-shareable	Outer Shareable	Outer Shareable
	Inner Shareable	Inner Shareable
	Non-shareable	Non-shareable

9.2.3 MPU register access

The MPU base and limit registers can be accessed indirectly or directly.

Indirectly

A region is selected by writing to the PRSEL_R_EL1 (PRSEL_R_EL2 for EL2 MPU). The selected region is programmed by writing to the PRBAR_EL1 and PRLAR_EL1 (PRBAR_EL2 and PRLAR_EL2 for EL2 MPU).

Directly

You can directly access only a group of 16 MPU region registers at a time by setting the PRSEL_R_EL_x.REGION[7:4].

If the Cortex®-R82 processor is implemented with 32 MPU regions, then you can directly access the first group of 16 MPU region registers from 0-15 by setting the PRSEL_R_EL_x.REGION[7:4] to 0b0000. Then you can directly access the second group of 16 MPU region registers from 16-31 by setting the PRSEL_R_EL_x.REGION[7:4] to 0b0001. You cannot access both groups of MPU region registers directly at the same time.

The base and limit registers for region *n*, where *n* is 0-15, are directly accessed by encoding the region number into CR_m and opcode2 of the following system register access instructions:

CR_m = 0b1rrr, where rrr = region_number[3:1].

op2 = 0brr00 for PRBAR_EL_x where *r* is *n*[0] and *n* is region_number 0-15 for the currently enabled group.

op2 = 0brr01 for PRLAR_EL_x where *r* is *n*[0] and *n* is region_number 0-15 for the currently enabled group.

Writing base and limit registers:

```

PRBAR0_EL1 MSR PRBAR<n>_EL1, <Xt>
-
PRBAR15_EL1
PRLAR0_EL1 MSR PRLAR<n>_EL1, <Xt>
-
PRLAR15_EL1
PRBAR0_EL2 MSR PRBAR<n>_EL2, <Xt>
-
PRBAR15_EL2
PRLAR0_EL2 MSR PRLAR<n>_EL2, <Xt>
-
PRLAR15_EL2

```

Reading base and limit registers:

```

PRBAR0_EL1 MRS PRBAR<n>_EL1, <Xt>
-
PRBAR15_EL1
PRLAR0_EL1 MRS PRLAR<n>_EL1, <Xt>
-
PRLAR15_EL1
PRBAR0_EL2 MRS PRBAR<n>_EL2, <Xt>
-
PRBAR15_EL2
PRLAR0_EL2 MRS PRLAR<n>_EL2, <Xt>
-
PRLAR15_EL2

```

The Cortex®-R82 Processor will dual issue MSR or MRS instructions accessing these registers under the following conditions:

- Both MSR or both MRS
- <N> is the same for both instructions
- Both registers are EL1 or both registers are EL2

9.3 MMU

The *Memory Management Unit* (MMU) is responsible for translating addresses of code and data *Virtual Addresses* (VAs) to *Physical Addresses* (PAs) in the real system. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

The three main functions of the MMU are to:

- Control the table walk hardware that accesses translation tables in main memory.
- Translate *Virtual Addresses* (VAs) to *Physical Addresses* (PAs).
- Provide fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables.

Each stage of address translation uses a set of address translations and associated memory properties that are held in memory mapped tables that are called translation tables. Translation table entries can be cached into a *Translation Lookaside Buffer* (TLB).



Pagetales must be placed on the *Main Manager* (MM) port. A fault is raised if a pagewalk occurs to a different port.

The following table describes the MMU components.

Table 9-6: MMU components

Component	Description
L1 instruction TLB	15 entries, fully associative
L1 data TLB	16 entries, fully associative
L2 TLB	1024 entries, 4-way set associative
Walk cache RAM	32 entries, 4-way set associative

A TLB entry refers to the information needed to perform a translation for a single page. The TLB entries contain either one or both of a global indicator and an *Address Space Identifier* (ASID) to allow context switches without requiring the TLB to be invalidated.

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



The Cortex®-R82 processor is optimized for 16KB or larger pages. There may be a performance impact if 4KB pages are used in VMSA.

The Cortex®-R82 processor has a physical address width that is parameterized and can be either 40 (when parameter PA_W=40) or 48 (when parameter PA_W=48) which allows 1TB and 256TB of physical memory, respectively, to be addressed.

9.3.1 TLB organization

The *Translation Lookaside Buffer* (TLB) is a cache of recently executed page translations within the *Memory Management Unit* (MMU).

The Cortex®-R82 processor implements a two-level TLB structure. The L2 TLB stores all page sizes and is responsible for breaking these down into smaller pages when required for the data side or instruction side L1 TLB.

TLB lockdown is not supported.

After reset, an Invalidate All operation is executed and all entries in the TLB are invalidated.

9.3.1.1 L1 TLB

The first level of caching for the translation table information is an L1 *Translation Lookaside Buffer* (TLB), implemented on each of the instruction and data sides.

The Cortex®-R82 L1 instruction TLB supports 4KB pages only for a single entry.

The Cortex®-R82 L1 data TLB supports 4KB pages only for a single entry.



L1 instruction TLB and L1 data TLB can still hold 16KB or larger pages as multiple entries. For example, an 16KB page is held as four 4KB entries. There is no performance impact with pages larger than 4KB.

Any other page sizes are fractured after the L2 TLB and the appropriate page size sent to the L1 TLB.

All TLB maintenance operations affect both the L1 instruction and data TLBs and cause them to be invalidated.

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

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

9.3.1.2 L2 TLB

A unified L2 *Translation Lookaside Buffer* (TLB) handles any misses from the L1 instruction and data TLBs.

The L2 TLB is a 4 way set-associative with 256 sets. Therefore it can cache up to 1024 translation results. The L2 TLB supports all *Virtual Memory System Architecture* (VMSA) block sizes, except for 1GB. See VMSAv8 in the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

A miss in the L1 data TLB or a hit in the L2 TLB has a penalty compared to a hit in the L1 data TLB. This penalty can be increased depending on the arbitration of pending requests. The best case hit penalty is 4 cycles and the worst case hit penalty is 9 cycles.

If a 1GB block is fetched, it is split into 512MB blocks and the appropriate block for the lookup is stored.

Accesses to the L2 TLB take a variable number of cycles, based on:

- Competing requests from the L1 TLBs.
- TLB maintenance operations in flight.
- Different page size mappings in use.

9.3.1.3 Walk cache RAM

The walk cache RAM holds the result of a stage 1 translation up to, but not including, the last level.

9.3.2 TLB match process

The Arm®v8-R AArch64 architecture provides support for multiple *Virtual Address* (VA) spaces that are translated differently.

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

Each TLB entry contains a:

- VA.
- *Physical Address* (PA).
- Set of memory properties that includes type and access permissions.

Each entry is either associated with a particular *Address Space Identifier* (ASID) or is global. In addition, each TLB entry contains a field to store the *Virtual Machine Identifier* (VMID) in the entry applicable to accesses from Non-secure EL0 and EL1 Exception levels.

Each TLB entry is associated with a particular translation regime:

- EL0 or EL1 in Secure state

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

- When VA[48:N] matches the requested address, where N is \log_2 of the block size for that translation that is stored in the TLB entry, moderated by the page size.
- The ASID matches the current ASID held in the CONTEXTIDR, TTBRO, or TTBR1 register, or the entry is marked global.
- The VMID matches the current VMID held in the VTTBR_EL2 register.

9.3.3 Translation table walks

When an access to an address is requested, 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 translation proceeds by looking up the translation table during a translation table walk.

When the Cortex®-R82 processor generates a memory access, the MMU:

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

In the case of an L2 TLB miss, the hardware does a translation table walk as long as the MMU is enabled and the translation using the base register has not been disabled.

If the translation table walk is disabled for a particular base register, the Cortex®-R82 processor returns a translation fault.

If the TLB finds a matching entry, it checks the access permission bits and the domain to determine if the access is permitted. If the matching entry does not pass the permission checks, the MMU signals a permission fault.

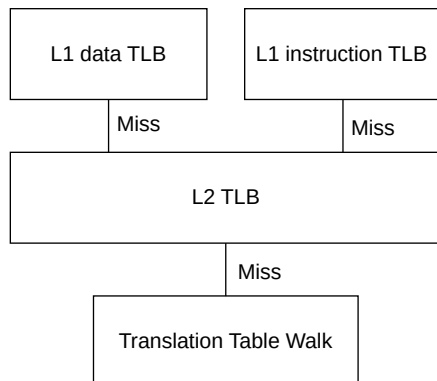
See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for details of Permission faults, including:

- A description of the various faults
- The fault codes.
- Information regarding the registers where the fault codes are set.

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

The following figure shows the translation table walk process:

Figure 9-3: Translation table walks



9.3.4 MMU memory accesses

During a translation table walk, the *Memory Management Unit* (MMU) generates accesses. This section describes the specific behaviors of the Cortex®-R82 processor for MMU memory accesses.

9.3.4.1 Configuring MMU accesses

Translation table walk can be performed in Cacheable or Non-cacheable regions. This is determined by the translation table walk memory attributes, which can be affected by several different configurations:

- IRGN and ORGN bits in the TCR_EL1 registers which define the memory type for translation table walk.
- SCTLR_ELx.C and HCR_EL2.CD which affect the table walk to Cacheable or Non-cacheable memory.
- Stage 2 memory attributes for stage 1 translation table walk, which affect the stage 1 translation table walk memory attribute.

Only when the final translation table walk memory attribute is Inner Write-Back and Outer Write-Back and the cache is enabled, the translation table walk accesses the cacheable memory.

For more information on the control fields, see the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

9.3.4.2 Hardware management of the Access flag

The Cortex®-R82 processor includes the option to perform hardware updates to the translation tables.

This feature is enabled in register TCR_EL1.

The Cortex®-R82 processor supports hardware updates to the Access flag only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. The Cortex®-R82 processor does not support hardware management of dirty state.

If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the Cortex®-R82 processor returns an abort with the following encoding:

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

For more information about hardware updates of the Access flag, see the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

9.3.5 Responses

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

9.3.5.1 MMU responses

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

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

The response from the MMU contains the following information:

- The *Physical Address* (PA) corresponding to the translation.
- A set of permissions.
- Secure or Non-secure state information.



The Secure or Non-secure state information is the security state of the descriptor and not the security state of the Cortex®-R82 processor. The Cortex®-R82 processor always operates in Secure state but can access both Secure and Non-secure physical memory address space.

- All the information that is required to report aborts.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more details.

9.3.5.2 MMU aborts

The *Memory Management Unit* (MMU) can detect faults that are related to address translation and can cause exceptions to be taken to the individual core within the Cortex®-R82 processor.

Faults can include address size, translation, access flags, and permissions. See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information about aborts.

9.3.5.3 External aborts

External aborts are aborts that occur in the memory system that are different than the *Memory Management Unit* (MMU) detects. External aborts are caused by errors flagged by the external memory interfaces or are generated because of an uncorrected ECC error in the L1 data cache or L2 cache arrays.

When an External abort to the external interface occurs on a translation table walk access, the MMU returns a synchronous External abort. For a Load pair or a Store pair operation, the address captured in the fault register is that of the address that generated the synchronous external abort.

9.3.5.4 Misprogramming contiguous hints

A programmer might misprogram the translation tables so that:

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

If there is this kind of misprogramming, the Cortex®-R82 processor does not generate a translation fault.

9.3.5.5 Conflict aborts

Conflict aborts are generated from the L1 instruction or data *Translation Lookaside Buffer* (TLB). If a conflict abort is detected in the L2 TLB, it chooses one valid translation. The L2 TLB does not generate a conflict abort.

9.3.6 Memory behavior and supported memory types

The Cortex®-R82 processor support all the Arm®v8-R AArch64 memory types.

These device memory types have the following three 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 acknowledge of transactions from the interconnect.

The permitted combinations are described in the following table:

Table 9-7: Supported Arm®v8-R AArch64 Device memory types

Memory type	Description
Device-GRE	Device Gathering, Reordering, Early Write Acknowledgement. Device-GRE is similar to Normal Non-cacheable but does not permit Speculative accesses.
Device-nGRE	Device non-Gathering, Reordering, Early Write Acknowledgement. Transactions might be reordered within the L2 memory system or in the system interconnect. The use of barriers is required to order accesses to Device-nGRE memory.

Memory type	Description
Device-nGnRE	Device non-Gathering, non-Reordering, Early Write Acknowledgement. Device-nGnRE is equivalent to the Device memory type in earlier versions of the architecture. Device-nGnRE is treated the same as nGnRnE inside the Cortex®-R82 processor but is reported differently on the bus interface.
Device-nGnRnE	Device non-Gathering, non-Reordering, No Early Write Acknowledgement.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information.

Write-Through and Mixed inner and outer Cacheability not included.

9.3.7 Page-based hardware attributes

Page-Based Hardware Attributes (PBHA) is not supported by the Cortex®-R82 processor.

10. RAS Extension support

This chapter describes the *Reliability, Availability, and Serviceability* (RAS) features implemented in the Cortex®-R82 processor.

10.1 RAS Extension support in the processor

The Cortex®-R82 processor reports and records errors in memories and system ports according to the *Reliability, Availability, and Serviceability* (RAS) Extension for the Arm®v8.4 architecture.

The RAS Extension is always present in the Cortex®-R82 processor, however most of the features are only supported when RAM protection is configured.

If RAM protection is configured, all memory errors are recorded in the appropriate RAS registers unless all such Error Record registers are already full with errors of higher severity. The Cortex®-R82 processor records all memory errors regardless of whether they are encountered during speculative execution or not. Errors are recorded both by the node that detects them (for example, a first cache discovering a corrupted line and poisoning the location) and by the node that consumes them (for example, a second cache snooping the first cache for a critical read). Bus protection is not available in the Cortex®-R82 processor.



Errors detected on speculative accesses are recorded but no exceptions (synchronous or asynchronous) are generated. If the Cortex®-R82 processor attempts to consume a poisoned location later, an exception will be taken at that time.

In particular, the Cortex®-R82 processor RAS implementation supports:

- Cache protection with *Single Error Correct Double Error Detect* (SECCDED) on the RAMs that contain dirty data or coherency state. This includes the *Instruction Tightly Coupled Memory* (ITCM), *Data Tightly Coupled Memory* (DTCM), RAMs, L1 data cache data and dirty RAMs, L2 cache tag and data RAMs, L2 cache data buffers, L2 duplicate L1 tag RAMs, and *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs.
- Cache protection with *Double Error Detect* (DED) on the RAMs that only contain clean data or, in the case of the L1 data cache tag RAM, data that can be corrected via the duplicate L1 tag RAMs. This includes the L1 instruction cache tag and data RAMs, L1 data cache tag RAM, and the L2 *Translation Lookaside Buffer* (TLB) tag and data RAMs.
- The *Error Synchronization Barrier* (ESB) instruction. When an ESB instruction is executed, the core ensures that all unrecoverable SErrors Interrupts that are generated by instructions before the ESB are either taken by the core or pended in DISR_EL1.
- The *implicit* ESB instruction as described in the Arm®v8 architecture RAS extension.
- Poison attribute on bus transfers.
- Core and cluster Error Record registers.

- *Fault Handling Interrupts* (FHIs).
- *Error Recovery Interrupts* (ERIs).
- *Critical Error Interrupt* (CEI).
- Error injection.

The L2 cache replacement RAMs and the branch predictor RAMs are out of the scope of RAS, as they are protected by alternate means. The Cortex®-R82 processor remains robust in the presence of any errors in these RAMs as well.

If RAM protection is not configured, the Cortex®-R82 processor has one RAS node, Node 0, and one Error Record register, Error Record register 0, implemented. Node 0 is for the shared memory, non-ECC errors from the L2 cache and LCU. This node corresponds to Error Record register 0.

If RAM protection is configured, the Cortex®-R82 processor adds two more RAS nodes, Node 1 and Node 4, and six more Error Record registers, Error Record registers 1 to 6:

- Node 1 for per-core memory *Error Correcting Code* (ECC) errors from the L1 data cache RAMs, L1 instruction cache RAMs, *Tightly Coupled Memories* (TCMs) (ITCM and DTCM), and the L2 TLB RAMs. This node corresponds to Error Record registers 1-3.
- Node 4 for shared memory ECC errors from the L2 cache data RAMs, L2 cache tag RAMs, L2 cache data buffer RAMs, L2 cache duplicate L1 tag RAMs, and the LCU duplicate L1 tag RAMs. This node corresponds to Error Record registers 4-6.

The following table shows per-core and shared nodes and error record registers in the Cortex®-R82 processor.

Table 10-1: RAS nodes and error record registers

Node	Error Record register	Error types recorded	Usage	RAS node control
0	0	Non-ECC memory errors from L2 cache and LCU	Shared	ERR0CTLR
1	1, 2, 3	Memory errors from L1 instruction cache, L1 data cache, ITCM, DTCM, L2 TLB	Per-core	ERR1CTLR
4	4, 5, 6	Memory errors from L2 cache and LCU	Shared	ERR4CTLR

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

10.2 Memory protection behavior

The Cortex®-R82 processor is robust in the presence of 1-bit or 2-bit errors in any of its RAMs.

The Cortex®-R82 processor employs a variety of error detection and correction schemes. *Single Error Correct Double Error Detect* (SECEDED) *Error Correcting Code* (ECC) is used if there is a need

to produce corrected data from the error data. *Double Error Detect* (DED) ECC is used when error detection is sufficient and the corrected data is obtained from a different processor memory. Cortex®-R82 also makes use of non-RAM redundancy or functional correctness by construction to protect from 1-bit and 2-bit errors in the branch predictor and L2 replacement RAMs. For example, errors in the branch predictor RAMs cannot cause a functional issue - a mispredicted branch as a result of an error will be resolved and the pipeline will be flushed and the correct instruction fetched. 1-bit and 2-bit errors are corrected automatically by design. Equally the L2 replacement RAM is used to improve cache eviction policy; errors will only result in a different victim being selected and execution will continue as normal.

In the Cortex®-R82 SECEDED and DED schemes, when the datum and code bits are all-zero, or all-one, the interpretation is that an error has occurred that the ECC scheme cannot correct. When the datum and code bits are both located within a single physical RAM instance, RAM failure modes that results in all-zero or all-one output will be detected as a fatal error. However in some cases it may still be possible to correct, such as when the data is known to be clean and can be invalidated and refetched.

When the datum and code bits are both located within a single physical RAM instance, RAM failure modes that results in all-zero or all-one output will be detected as a fatal error.

Regardless of the state of the processor, the Cortex®-R82 processor records the error at the point when it is detected.

When the Cortex®-R82 processor is operating in Lock or Hybrid mode, the primary cluster or core level memories are used to provide responses to the redundant cores. This improves availability by ensuring that error detection and correction apply to the lockstep pair.

The following table indicates which protection type is applied to each RAM. The ECC granule specifies how much data is used to compute a single ECC code.

Table 10-2: RAM protection

RAMs	Protection scheme	Error type	Action
Instruction Tightly Coupled Memory (ITCM)	SECEDED 64 bits	1-bit	Correct the error
		2-bit	Poison RAM location
Data Tightly Coupled Memory (DTCM)	SECEDED 32 bits	1-bit	Correct the error
		2-bit	Poison RAM location
L1 instruction cache data	DED 64 bits	1-bit/2-bit	Evict cache line and refetch clean line from L2 cache, LLRAM, or MM
L1 instruction cache tag	DED 30 bits ¹	1-bit/2-bit	Evict cache line and refetch clean line from L2 cache, LLRAM, or MM

¹ Granule size is configuration dependent.

RAMs	Protection scheme	Error type	Action
L1 data cache data	SECDED 32 bits	1-bit	Evict cache line, correcting the error when needed, and refetch clean line from L2 cache, LLRAM, or MM.
		2-bit	Poison the location or evict poison to the L2
L1 data cache tag	DED Up to 31 bits ²	1-bit/2-bit	Evict cache line and refetch clean line, retrieving correct tags from the L2/LCU duplicate L1 tag memories
L1 data cache dirty	SECDED 2 bits	1-bit	Correct the error
		2-bit	Assume cache line is clean, dirty data discarded
L2 cache data	SECDED 128 bits	1-bit	Correct the error
		2-bit	Poison the location
L2 cache tag	SECDED Up to 33 bits ³	1-bit	Correct the error
		2-bit	The L2 Cache provides the IMP_CLUSTERACTLR_EL1.L2ECHLT software control that enables additional error containment support in the event of detecting an uncontrollable error in the L2 Cache or L2 Duplicate L1 tags. When enabled, a fatal error in the tag RAMs will prevent the instruction from being replayed and instead stall all subsequent coherent accesses to the L2 Cache and L2 Duplicate L1 Tag RAMs. This will still cause failure of the affected execution thread(s), but it will prevent the consumption of corrupt state without any error indication. The stall behavior will remain until reset of the L2. Note that if the L2 is configured with L2_SLICES=2 each L2 slice will be stalled independently as the RAMs are split between the two slices. MBIST requests will still be allowed to access the RAMs when the L2 has entered this stall condition.
L2 cache data buffers	SECDED 144 bits	1-bit	Correct the error
		2-bit	Poison the location
L2 duplicate L1 tag	SECDED Up to 32 bits ⁴	1-bit	Correct the error
		2-bit	Evict cache line and refetch clean line from MM, unless location is about to be overwritten, or optionally stall the request
LLRAM Coherency Unit (LCU) duplicate L1 tag	SECDED Up to 18 bits ⁵	1-bit	Correct the error
		2-bit	Evict cache line and refetch clean line from LLRAM
L2 cache replacement	Functional correctness by construction	1-bit/2-bit	Cache evicts a line that might be different compared with the replacement policy. No error results from selecting a different line. Errant entry replaced with correct data.
Branch predictor	Non-RAM redundancy	1-bit/2-bit	Error data checked and corrected by core pipeline
L2 Translation Lookaside Buffer (TLB) data	DED 46 bits	1-bit/2-bit	Discard the translation and repeat the page walk

² Granule size is configuration dependent.³ Granule size is configuration dependent.⁴ Granule size is configuration dependent.⁵ Granule size is configuration dependent.

RAMs	Protection scheme	Error type	Action
L2 TLB tag	DED 68 bits	1-bit/2-bit	Discard the translation and repeat the page walk

Error correction

The Cortex®-R82 processor corrects all single bit errors either by correcting the error (inline correction for caches and TCMs, or overwriting with correct data for branch predictors and L2 cache replacement) or by evicting the related cache line and by re-fetching a clean copy from a different memory. Therefore, single bit errors do not affect the function of the Cortex®-R82 processor, although they might affect timing. If there are multiple single bit errors in different RAMs, or in different protection granules in the same RAM, then the Cortex®-R82 processor also remains functionally correct.

If there is a double bit error in a single RAM in a single protection granule, then the Cortex®-R82 processor detects the error. If the data can be found in a different memory (such as the clean data in a cache) or can be recomputed (such as branch outcomes or L2 cache replacement), the Cortex®-R82 processor attempts to correct the error by evicting the cache line and by re-fetching a clean copy from a different memory. If the data cannot be found elsewhere (such as the TCM data or dirty data in a cache), then the data is lost. In some cases, such as a double bit error on a tag RAM of a dirty cache line, the coherency state might be lost, for example because:

- The address of the erroneous cache line has become **UNKNOWN** and the location cannot be poisoned, or
- The coherent state of the line has been lost and it is no longer possible to determine if a line is clean or dirty.

In addition to correcting the data read from protected RAMs, the Cortex®-R82 processor also either writes the corrected data back into the protected RAMs or it invalidates the erroneous location or poisons the erroneous location. This allows it to avoid the accumulation of errors that leads to correctable errors combining into uncorrectable errors or double bit errors combining into triple bit errors.

If there are three or more bit errors in the same protection granule, then depending on the RAM and the position of the errors in the RAM, the Cortex®-R82 processor might or might not detect the errors.

The memory protection feature of the Cortex®-R82 processor has a minimal performance impact when no errors are present.

The Cortex®-R82 processor also includes hard error handling mechanisms to guarantee forward progress in the presence of a limited number of hard correctable errors. A hard error is a physical error in the RAM that prevents the correct value being written, for example, due to a permanently stuck-at bitcell.

A single hard error can be corrected and is guaranteed to make progress. However, if there are multiple hard errors then in some cases this can cause live-locks as the line could continuously replay.

The following table describes the hard error avoidance mechanisms for the Cortex®-R82 RAMs.

Table 10-3: RAM hard error avoidance mechanisms

RAMs	Mechanism
<i>Instruction Tightly Coupled Memory (ITCM)</i>	Single 128-bit correction buffer acts as Write-Through cache
<i>Data Tightly Coupled Memory (DTCM)</i>	Single 128-bit correction buffer acts as Write-Through cache
L1 instruction cache data	Cache line has been invalidated; replayed access treated as Non-cacheable
L1 instruction cache tag	Cache line has been invalidated; replayed access treated as Non-cacheable
L1 data cache data	<i>Cache Line Avoidance Register (CLAR)</i> to mask hits and allocations on most recently affected cache line
L1 data cache tag	CLAR to mask hits and allocations on most recently affected cache line
L1 data cache dirty	CLAR to mask hits and allocations on most recently affected cache line
L2 cache data	Correct inline to allow current request to proceed
L2 cache tag	Replayed access streams from the tag write buffer to avoid the cache RAM
L2 cache data buffers	Correct inline to allow current request to proceed
L2 cache replacement RAMs	Request proceeds by evicting a cache line that might be different compared with the replacement policy
L2 duplicate L1 tag	Replayed access streams from the tag write buffer to avoid the RAM
<i>LLRAM Coherency Unit (LCU) duplicate L1 tag</i>	Correct in-line or invalidate and allow current request to proceed
Branch predictor	Execution proceeds from the correct branch target
L2 <i>Translation Lookaside Buffer (TLB)</i> data	CLAR to mask affected data
L2 TLB tag	CLAR to mask affected data

The CLAR is a dedicated structure that caches the most recently corrected error. Future requests to the same location use the CLAR contents instead of reading again from the affected RAM.

10.3 Error containment

The Cortex®-R82 processor is designed to maximize error containment, which makes sure that an error is detected and not silently propagated.

Error containment also implies support for poisoning to ensure that errors are reported only when the erroneous data is consumed.

Support for the *Implicit Error Synchronization Barrier (IESB)* also allows further isolation of imprecise exceptions and future operations that are reported when poisoned data is consumed.



Using IESB might affect your interrupt latency response. For more information on interrupt latency, see [8.14.1 Interrupt latency](#) on page 183.

The Cortex®-R82 processor avoids Uncontainable errors with the following exceptions:

- A fatal double bit error on the L1 data cache dirty RAM. When a snoop detects a fatal dirty RAM error, it cannot determine whether the line is clean or dirty. The data is assumed to be clean since the coherent state is unknown, and therefore might not drain data to the L2 cache. An Uncontainable error is raised to the RAS node associated with the core.
- When an uncorrectable error is detected in an L2 cache data RAM, the chunk of data with the error is marked as poisoned. The poison is stored per 128-bits of data. If the interconnect supports poisoning, the poison is passed along with the data when the line is evicted or snooped from the cluster and a Deferred error is reported to the RAS node. No abort is generated when a line is poisoned, as the abort can be deferred until the point when the poisoned data is consumed.

If a poisoned cache line is evicted or snooped from the cluster and the interconnect does not support poisoning the Cortex®-R82 processor generates an interrupt `nCOMPLEXCRITIRQ` and reports an Uncontainable error to the RAS node before transferring the data, since the poison state is lost. Software can indicate if the interconnect supports poisoning or not by setting the interconnect data poisoning bit in the cluster control register.

- The *L2 Data Buffers* (L2DB) are used for temporary storage of data within the L2, and can forward data on to the L2 cache, the L1 memory system or the *Main Accelerator Coherency Port* (MACP). When a read-modify-write is performed on an L2DB and an uncorrectable error is detected the L2DB will be marked as poisoned. The poison is stored per 128-bits of data. In all other cases the L2DB itself is not poisoned; if the data request is for an L1, the *Main Manager* (MM) port or the MACP then poison will be transferred alongside the data to the destination, while an L2 cache allocation from the L2DB will transfer the unmodified ECC bits directly into the L2 cache. A subsequent access to the L2 cache will detect the uncorrectable error and poison the location. In all cases these errors will be reported Deferred error to the RAS node. No abort is generated when a line is poisoned, as the abort can be deferred until the point when the poisoned data is consumed.

If an uncorrectable error is detected in the L2DBs then the byte strobes associated with the write may be lost. The poisoned write propagated on the external interface in this scenario may access bytes not written by the original request in cases where a partial poison granule is being written.

If an uncorrectable error is transferred from the L2DB to the MM port and the interconnect does not support poisoning the Cortex®-R82 processor generates an interrupt `nCOMPLEXCRITIRQ` and reports an Uncontainable error to the RAS node before transferring the data, since the poison state is lost. Software can indicate if the interconnect supports poisoning or not by setting the interconnect data poisoning bit in the cluster control register.

- A fatal double bit error on the L2 cache tag RAM. The hit/miss status and address associated with the cache line is unknown. To proceed, the line is invalidated in the L2 cache and the access is replayed. An Uncontainable error is raised to the RAS node associated with the cluster.
- A fatal double bit error on the L2 duplicate L1 tag RAM. This hit/miss status and address associated with the cache line is unknown. To proceed, the line is invalidated in both the L2 duplicate L1 tag RAM and the corresponding L1 cache and the access is replayed. An Uncontainable error is raised to the RAS node associated with the cluster.

- An uncorrectable error occurred in any RAM that cannot be associated with a specific instruction (typically an already retired store). The error is considered to be Uncontainable because later instructions in the program order may have consumed corrupted architectural state.

Uncontainable errors can lead to **UNPREDICTABLE** behavior. They can result in further data corruption, or in software algorithms deadlocking as they can read inconsistent data. Therefore it is generally not possible to cleanly recover from such errors. Arm recommends that in response to nCOMPLEXCRITIRQ interrupts a system reset is performed as soon as possible.

When a dirty cache line with an error on the data RAMs is evicted from the core, the write on the MM interface still takes place. However, if the error is uncorrectable then:

- On AXI5, the uncorrected data is written.
- On CHI, the uncorrected data is written but the data poison field indicates that there is a data error.
- On AXI5, the nCOMPLEXCRITIRQ pin is asserted.



The snoop response does not indicate the error.

-
- On CHI, the snoop response indicates that either the data is poisoned (if supported), or that there is an error.

In some cases, it is possible for an error to be counted more than once. For example, multiple accesses might read the location with the error before the line is evicted.

10.4 Fault detection and reporting

When the Cortex®-R82 processor detects a fault, it raises a *Fault Handling Interrupt* (FHI) exception through the fault signals. FHIs are reflected in the Error Record registers that are updated in the node that detects the errors.

Fault handling interrupts

When ERROCTL.FI is set, all detected non-ECC Deferred errors and Uncorrected errors that the cluster detects in the L2 cache and *LLRAM Coherency Unit* (LCU) generate an FHI on the nCOMPLEXFAULTIRQ signal.

When ERR1CTL.FI is set, all detected *Error Correcting Code* (ECC) Deferred errors and Uncorrected errors that core n detects in the L1 instruction, L1 data, and TCM RAMs generate an FHI on the nCOREFAULTIRQ[n] signal.

When ERR4CTL.FI is set, all detected ECC Deferred errors and Uncorrected errors that the cluster detects in the L2 cache tag RAMs, L2 cache data RAMs, and LCU duplicate L1 tag RAMs generate an FHI on the nCOMPLEXFAULTIRQ signal.

When `ERR0CTLR.CFI`, `ERR1CTLR.FI`, or `ERR4CTLR.CFI` or any other CE-counter overflow bits are set, then all detected Corrected errors also generate a core or cluster FHI respectively.

Error recovery interrupts

When `ERR0CTLR.UI` is set, all detected non-ECC Uncorrected errors that are not deferred that the cluster detects generate an error recovery interrupt on the `nCOMPLEXERRIRQ` signal.

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

When `ERR4CTLR.UI` is set, all detected ECC Uncorrected errors that are not deferred that the cluster detects generate an error recovery interrupt on the `nCOMPLEXERRIRQ` signal.

Critical error interrupts

When `ERR0CTLR.CI`, `ERR1CTLR.CI`, or `ERR4CTLR.CI` is set, all critical errors that the Cortex®-R82 processor detects generate a critical error interrupt on the `nCOMPLEXCRITIRQ` signal. All Uncontainable error cases mentioned in [10.3 Error containment](#) on page 223 generate a critical error interrupt.

Clearing reported faults

The `nCOREFAULTIRQ`, `nCOREERRIRQ`, `nCOMPLEXFAULTIRQ`, `nCOMPLEXERRIRQ`, and `nCOMPLEXCRITIRQ` signals remain asserted until software clears them by writing the `ERR<n>STATUS` register.

Arm recommends that at least the `nCOMPLEXERRIRQ`, `nCOMPLEXCRITIRQ`, and `nCOREERRIRQ` pins are connected to the interrupt controller, so that an interrupt or system error is generated when the pin is asserted.

10.5 Error detection and reporting

When the Cortex®-R82 processor consumes an error, it raises different exceptions depending on the error type.

The Cortex®-R82 processor might raise:

- A *Synchronous External Abort* (SEA) when the error that it detects is an architecturally executed instruction (typically a load) that has not retired.
- An *Asynchronous External Abort* (AEA), reported as an *SError Interrupt* (SEI) when the error it detects cannot be associated with a specific instruction (typically an already retired store). The error is Uncontainable, because it is reported asynchronously and the Cortex®-R82 processor cannot guarantee that subsequent instructions have not consumed corrupted data.
- A *Fault Handling Interrupt* (FHI) when it detects an uncorrectable, correctable, or deferred error through the `nCOREFAULTIRQ` signal if the error is in the core RAMs or `nCOMPLEXFAULTIRQ` if the error is in the cluster RAMs.

- An *Error Recovery Interrupt* (ERI) when it detects an uncorrectable error through the nCOREERRIRQ signal if the error is in the core RAMs or nCOMPLEXERRIRQ if the error is in the cluster RAMs.
- A *Critical Error Interrupt* (CEI) when it detects an uncontrollable error in either the core or the cluster RAMs through the nCOMPLEXCRITIRQ signal.

Error reporting registers

The errors detected in the following memories, including corrected and uncorrected errors, are logged in the ERR<n>STATUS and ERR<n>MISCO registers:

- *Instruction Tightly Coupled Memory* (ITCM) and *Data Tightly Coupled Memory* (DTCM) RAMs.
- L1 instruction cache tag and data RAMs.
- L1 data cache tag and data RAMs.
- L1 data cache dirty RAMs.
- L2 cache tag and data RAMs.
- L2 cache data buffers.
- L2 duplicate L1 tag RAMs.
- *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs.
- *L2 Translation Lookaside Buffer* (TLB) cache data and tag RAMs.

The ERR<n>STATUS registers indicate:

- If the error is corrected or uncorrected.
- If the error is deferred or reported.
- The type of memory where the error occurred.

The ERR<n>MISCO register indicates:

- The way, the index, the level, and the bank of the memory where the error occurred. This information identifies the RAM and the line within the RAM that contains the error.
- The corrected error counts. One counter is incremented for each corrected error detected.

Error reporting

Any detected error is reported in the *Error Record Primary Status Register* (ERR<n>STATUS), and the *Error Record Miscellaneous Register 0* (ERR<n>MISCO). Errors reported include errors that are successfully corrected and errors that cannot be corrected. If multiple errors occur on the same clock cycle, the Cortex®-R82 processor serializes them and records them separately. In the unusual case when multiple errors from multiple RAMs occur in consecutive clock cycles and the Cortex®-R82 processor cannot record them in a timely manner, the error type is set appropriately to indicate that multiple errors have been observed.



Simultaneous errors in different RAMs get serialized and reported but multiple errors within a single RAM such as multiple DTCM errors, are combined into a suitable multiple error report.

If the Cortex®-R82 processor is implemented without RAM protection, then, there is one Error Record register, Error Record register 0, provided. If the Cortex®-R82 processor is implemented with RAM protection, there are seven Error Record registers, Error Record registers 0 to 6, provided which can be selected with the ERRSELR_EL1 register:

- Error Record register 0 is for the cluster and is shared between all cores in the cluster. It records any non-ECC error in the L2 cache and LCU.
- Error Record registers 1-3 are private to the core, and are updated on any *Error Correcting Code* (ECC) error in the core RAMs including L1 data cache RAMs, L1 instruction cache RAMs, TCMs (ITCM and DTCM), and the L2 TLB RAMs.
- Error Record registers 4-6 are for the cluster and are shared between all cores in the cluster. They record any ECC error in the L2 cache data RAMs, L2 cache tag RAMs, L2 cache data buffer RAMs, L2 cache duplicate L1 tag RAMs, and the LCU duplicate L1 tag RAMs.

10.5.1 Error reporting and performance monitoring

All detected memory errors trigger the MEMORY_ERROR event.

The MEMORY_ERROR event is:

- Counted by the PMU counters if it is selected and the counter is enabled.
- Connected to the *Embedded Trace Macrocell* (ETM) as external event number 28.

10.6 Error injection

Error injection involves inserting an error in the error detection logic to verify the reporting and recording structure. Error injection does not verify syndrome generation or error detection and correction hardware.

Error injection uses the ERR<n>PFGCDN_EL1 and ERR<n>PFGCTL_EL1 registers to insert errors.

The Cortex®-R82 processor can inject the following error types:

For Record 0 (shared memory non-ECC errors): Error injection is not supported.

For Records 1-3 (per-core memory *Error Correcting Code* (ECC) errors):

- A *Corrected Error* (CE). This is reported as a single-bit ECC error on any L1 data cache access.
- A *Deferred Error* (DE). This is reported as a double-bit ECC error on any L1 data cache access.
- A *Recoverable Error* (UER) cannot be generated at this node.

- An *Unrecoverable Error* (UEU) cannot be generated at this node.
- An *Uncontainable Error* (UC). This is reported as a double-bit ECC error on an L1 data cache Main Memory eviction.

For Records 4-6 (shared memory ECC errors):

- A *Corrected Error* (CE). This is reported as a single-bit ECC error on any L2 cache data access.
- A *Deferred Error* (DE). This is reported as a double-bit ECC error on any L2 cache data access.
- A *Recoverable Error* (UER) cannot be generated at this node.
- An *Unrecoverable Error* (UEU) cannot be generated at this node.
- An *Uncontainable Error* (UC). This is reported as a double-bit ECC error on any L2 cache tag access.

An error can be injected immediately, subject to the triggering condition associated with each error type, or when a 32-bit counter reaches zero. You can control the value of the counter through the ERR<n>PFGCDN_EL1 register. The value of the counter decrements on a per clock cycle basis.



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

An example software sequence to insert an error could look like this:

1. Select the appropriate RAS node by programming ERRSELR_EL1
2. Enable error reporting by programming ERXCTLR_EL1
3. Write a countdown value to ERXPFGCDN_EL1
4. Select an error to be injected by programming ERXPFGCTL_EL1
5. Ensure the programming is applied by executing a DSB and then an ISB instruction
6. After the countdown to complete, error results should be visible by reading ERXSTATUS_EL1 and ERXMISCO_EL1

10.7 RAS register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary table for the Cortex®-R82 processor AArch64 *Reliability, Availability, and Serviceability* (RAS) registers in [A.1.7 AArch64 RAS registers summary](#) on page 297.

You can find the register summary table for the Cortex®-R82 processor external RAS registers in [B.1.1.1 External RAS registers summary](#) on page 1416.

11. GIC CPU interface

This chapter describes the Cortex®-R82 processor implementation of the Arm *Generic Interrupt Controller* (GIC) CPU interface.

11.1 About the GIC CPU interface

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

Each core within the Cortex®-R82 processor has a GIC CPU interface that includes registers to mask, identify, and control states of interrupts forwarded to a core. The GIC CPU interface, among other functions, turns external GIC distributor interrupt requests into IRQ and FIQ interrupts into the core.

A core accesses its GIC CPU interface registers via System register operations. Because the GIC CPU interface resides within the core, it is clock gated during WFI and WFE exceptions or placed into retention when the rest of the core enters retention. Any communication from the external GIC distributor causes a temporary wakeup or retention exit.

The Cortex®-R82 processor has a single shared pair of bidirectional AXI5-Stream interfaces that is internally connected to all cores. The Cortex®-R82 processor can be directly connected to an external GICv3 interrupt distributor within the system through AXI5-Stream interface.



The GIC AXI5-Stream interfaces operate in the same SCLK clock domain with the cluster. To support an integer ratio (for example, 2:1) for the GIC distributor, a clock enable signal ACLKENG is provided.

The external GIC distributor is responsible for receiving interrupts, prioritizing them and routing them to the associated interrupt input of the associated core within the Cortex®-R82 processor. The software is responsible for generating interrupts for inter-core communication by writing to the interrupt controller. Local peripherals, such as the timer, have dedicated interrupts inputs that are specific to the individual core.

The Cortex®-R82 processor conforms to the Arm GICv3.2 architecture. The Cortex®-R82 processor implements the GIC CPU interface as described in the *Arm® Generic Interrupt Controller Architecture Specification*. This chapter describes only features that are specific to the Cortex®-R82 processor implementation.

The Cortex®-R82 processor implementation of GICv3.2 architecture supports:

- Single Security state.
- Interrupt virtualization.
- *Software-generated Interrupts* (SGIs).

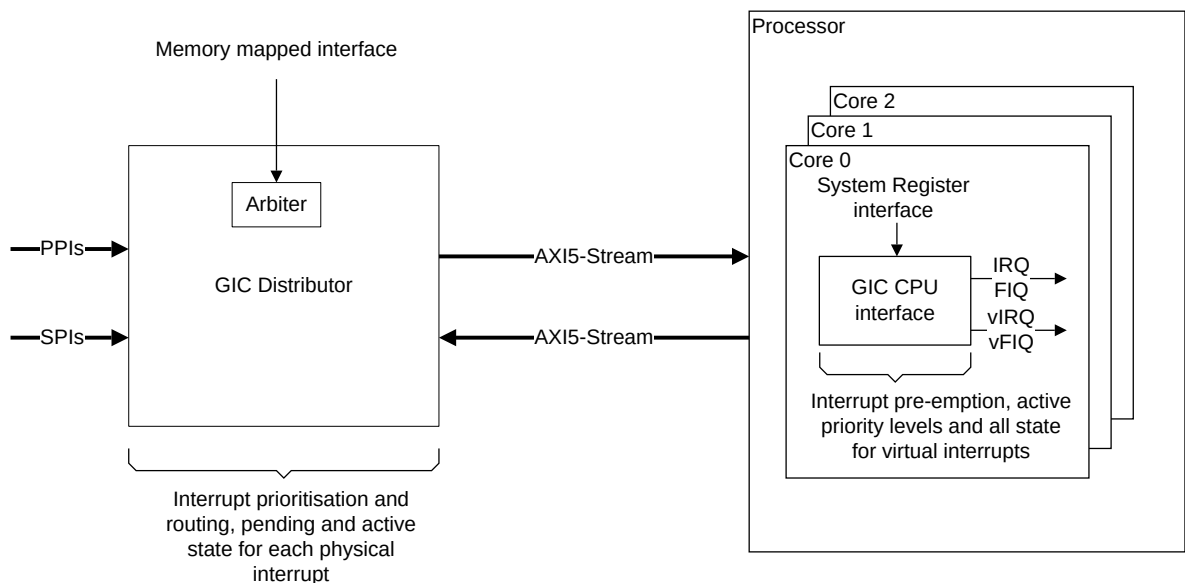
- Message Based Interrupts.
- System register access for the CPU interface.
- Interrupt masking and prioritization.
- Cluster environments, including systems that contain more than eight cores.
- Wake up events in power management environments.

The Cortex®-R82 processor GIC CPU interface includes interrupt grouping functionality that supports:

- Configuring each interrupt to belong to an interrupt group.
- Signaling Group 1 interrupts to the target core using the IRQ exception request only.
- 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.

The following figure shows GIC CPU interface.

Figure 11-1: GIC CPU interface



11.2 Disabling the GIC CPU interface

The Cortex®-R82 processor always includes the *Generic Interrupt Controller* (GIC) CPU interface, however you can disable it to meet your requirements.

You can disable the GIC CPU interface by asserting the GICCDISABLE signal HIGH at reset. Disabling the GIC CPU interface this way enables you to use other GIC architectures other than the GICv3.

If the Cortex®-R82 processor is not integrated with an external GICv3 interrupt distributor component in the system, then you need to disable the GIC CPU interface. If you disable the GIC CPU interface, then:

- The input signals nIRQ, nFIQ, nVIRQ, and nVFIQ can be driven by an external GIC in the system.
- GIC System register accesses generate **UNDEFINED** instruction exceptions.

11.3 Bypassing the GIC CPU interface

When the GIC CPU interface is enabled (GICCDISABLE signal LOW at reset), the Cortex®-R82 processor supports interrupt bypassing according to the GICv3 architecture.

You can use the ICC_SRE_EL2.DFB and ICC_IGRPEN0_EL1.Enable controls to bypass FIQ handling by the GIC. If you program the GIC CPU interface to bypass FIQ interrupts, the Cortex®-R82 processor routes the nFIQ and nVFIQ input signals directly to the CPU.



When the GIC CPU interface does not bypass IRQ interrupts, the input signals nVIRQ and nIRQ are ignored.

You can use the ICC_SRE_EL2.DIB and ICC_IGRPEN1_EL1.Enable controls to bypass IRQ handling by the GIC. If you program the GIC CPU interface to bypass IRQ interrupts, the Cortex®-R82 processor routes the nIRQ and nVIRQ input signals directly to the CPU.



When the GIC CPU interface does not bypass FIQ interrupts, the input signals nVFIQ and nFIQ are ignored.

For more information on programming the GIC CPU interface, see the [Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4](#).

11.4 GIC CPU interface register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary table for the Cortex®-R82 processor *Generic Interrupt Controller* (GIC) CPU interface registers in [A.1.6 AArch64 GIC system registers summary](#) on page 295.

12. Generic Timer

This chapter describes the Cortex®-R82 processor implementation of the Arm Generic Timer.

12.1 About the Generic Timer

The Generic Timer can schedule events and trigger interrupts based on an incrementing counter value. It generates timer events as active-LOW interrupt outputs and event streams.

The Cortex®-R82 processor Generic Timer is compliant with the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

This chapter describes only features that are specific to the Cortex®-R82 processor implementation.

12.2 Generic Timer functional description

The Cortex®-R82 processor provides a set of timer registers within each core in the cluster.

The timers are:

- An EL1 physical timer
- A Secure EL2 physical timer
- An EL1 virtual timer

The Cortex®-R82 processor does not include the system counter. This resides in the SoC. The system counter value is distributed to the Cortex®-R82 processor with a synchronous binary encoded 64-bit bus, CNTVALUEB[63:0].

When self-hosted trace is enabled, the system generic timer (physical or virtual) is used by the *Embedded Trace Macrocell* (ETM) for timestamp trace. When self-hosted trace is disabled, the CoreSight™ time value is identical to the system physical time value.

12.3 Generic Timer register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary table for the Cortex®-R82 processor Generic Timer registers in [A.1.10 AArch64 Generic Timer registers summary](#) on page 301.

13. Debug

This chapter describes the debug features of the Cortex®-R82 processor and the associated DebugBlock component.

13.1 About debug methods

The Cortex®-R82 processor supports two methods of invasive debugging to support software debugging, external debug and self-hosted debug. This section provides a brief introduction to these methods and outlines their main components.

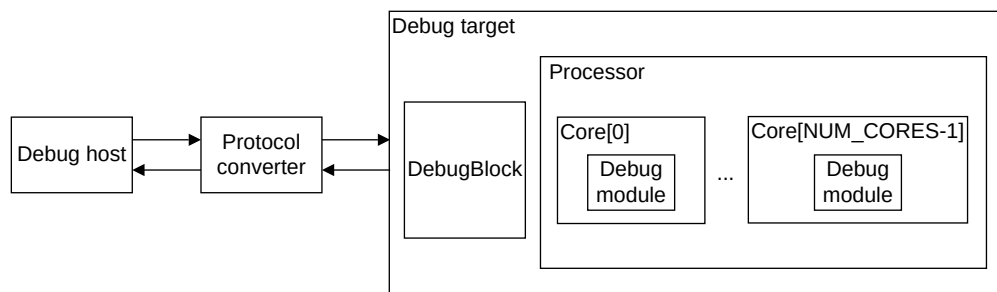
External debug

External debug is a conventional setup for debug, in which the debugger is external to the core that is being debugged. The debugger might be either hosted on another core within the Cortex®-R82 processor, or running external to the Cortex®-R82 processor. For example, the debugger might be hosted on a workstation connected using JTAG to a development system containing the Cortex®-R82 processor.

External debug is useful for hardware bring-up of a Cortex®-R82 processor-based system, that is, debugging during development when a system is first powered up and not all of the software functionality is available.

The following figure shows a typical external debug system.

Figure 13-1: External debug system



Debug host

The debug host is a computer, for example a personal computer, that is running a software debugger such as the DS-5 Debugger. With the debug host, you can issue high-level commands, such as setting a breakpoint at a certain location or examining the contents of a memory address.

Protocol converter

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

Debug target

The debug target is the lowest level of the system. An example of a debug target is a development system with a test chip or a silicon part with a Cortex®-R82 processor. The debug target implements system support for the protocol converter to access the Cortex®-R82 Debug modules.

Debug module

The debug modules within the Cortex®-R82 processor help with debugging software that is running on the processor such as an operating system or application software.

The debug modules enable an external debugger to:

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

Self-hosted debug

In self-hosted debug, the core being debugged within the Cortex®-R82 processor hosts debug monitor software itself. Hardware watchpoints and breakpoints generate debug exceptions on debug events.

Self-hosted debug is useful in situations in which the Cortex®-R82 processor has been deployed in a developed system, where there is no direct access to the Cortex®-R82 processor debug hardware. Self-hosted debug supports:

- Task debugging.
- OS and kernel debugging.
- Hypervisor debugging. Self-hosted debug support for Hypervisor code is limited to software breakpoint instructions.

For more information, see [13.4 Debug events](#) on page 241. These exceptions are handled by the debug monitor that typically resides alongside the operating system kernel or the hypervisor.

For details on self-hosted debug, see [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

13.2 Debug functional description

This section describes the trace, debug, and test features supported by the Cortex®-R82 processor. It includes Arm®v8-R AArch64 debug, cache debug, and CoreSight™ debug.

Arm®v8-R AArch64 debug architecture support

The Cortex®-R82 processor implements the Arm®v8-R AArch64 debug architecture including debug features up to Arm®v8.4 and the debug over powerdown Armv8.3-DoPD feature.

The Cortex®-R82 processor allows access to the internal debug functionality and registers either through a memory-mapped area on the external AMBA® APB5 completer port or by using System register operations from software running on the Cortex®-R82 processor.

Each core within the Cortex®-R82 processor implements six hardware breakpoints, four watchpoints, and a *Debug Communications Channel* (DCC). Four of the breakpoints match only against virtual address, the other two breakpoints match against either virtual address or context ID. All watchpoints can be linked to either of the virtual address or context-ID matching breakpoints to allow a memory request to be trapped in a given process context.

Cache debug

Cache debug allows software to read the content of the L1 cache, L2 cache, L2 duplicate L1 tag RAMs, and *LLRAM Coherency Unit* (LCU) duplicate L1 tag RAMs through **IMPLEMENTATION DEFINED** System registers.

See [8.12 Direct access to internal memories](#) on page 178 for more information on the mechanisms the Cortex®-R82 processor provides to read the internal memories.

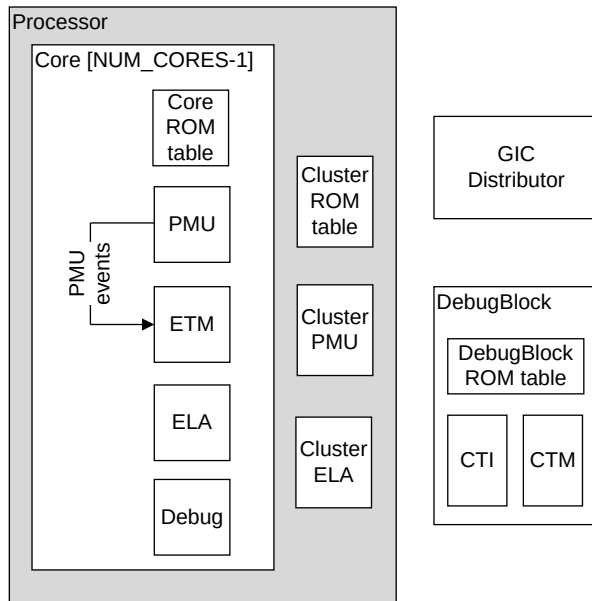
CoreSight™ debug

The Cortex®-R82 processor integrates several CoreSight™ debug related components to aid system debug along with CoreSight™ SoC.

These components include:

- Per-core *Embedded Trace Macrocell* (ETM).
- Per-core and a cluster level *Cross Trigger Interface* (CTI).
- *Cross Trigger Matrix* (CTM).
- Per-core and a cluster level *Performance Monitoring Unit* (PMU).
- Support for per-core and a cluster level CoreSight™ ELA-600 *Embedded Logic Analyzer*.
- Debug over powerdown support.
- Per-core ROM table, cluster ROM table, and DebugBlock ROM table.

The following figure shows the Cortex®-R82 processor CoreSight™ debug components.

Figure 13-2: Debug system components

The debug components are split into two groups. Some components are in the cluster itself and the rest are in a separate block named the DebugBlock. Separating the DebugBlock from the Cortex®-R82 processor allows you to put the DebugBlock in a separate power domain and place it physically with other CoreSight™ logic in the SoC, rather than close to the cluster. It also allows you to implement the debug components in an always on power domain, enabling debug over powerdown.

The connection between the cluster and the DebugBlock consists of a pair of APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. It includes register reads, writes, and CTI triggers. For more information, see [13.5 The DebugBlock](#) on page 242.

All debug components are controlled through the primary Debug APB interface on the DebugBlock. Requests on this bus are decoded by the APB decoder before being sent to the appropriate component in the DebugBlock or in the cluster. The per-core CTIs and the cluster CTI are connected to a CoreSight™ CTM.

Each core within the Cortex®-R82 processor contains an ETM, PMU, optional ELA, and debug component that are accessed using the debug APB bus. This block conforms to the Arm®v8-R AArch64 Debug Architecture Specification.

When the Cortex®-R82 processor is configured to include the optional ELA-600 instances, a number of key internal processor signals become observable. In the unlikely case where a processor bug is suspected, the Arm support team may provide programming sequences for ELA-600 and then analyze the captured data streams. Adding the optional ELA capabilities to the existing

Debug, PMU and Trace logic and combining triggers through the Cross-Trigger Matrix increase the likelihood of successfully debugging difficult and rare hardware issues.



The CoreSight ELA-600 is a separately licensable product.

The ETM in each core produces two separate instruction and data trace streams. The optional ELA in each core and the optional cluster ELA also produce data streams. All the core and cluster streams are combined and are output from the Cortex®-R82 processor using one AMBA® 4 ATB 32-bit interface (for ETM instruction trace streams) and one AMBA® 4 ATB 128-bit interface (for ETM data trace streams and ELA streams).

When an ELA is active, the relevant core or cluster clock will remain on and the associated logic will be unable to enter a low power state.

13.3 Debug register accesses

The Cortex®-R82 processor implements the Arm®v8-R AArch64 Debug architecture and debug events.

They are described in the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

The Debug architecture defines a set of debug registers. The debug registers can be accessed from:

- Software running on the core.
- An external debugger.

13.3.1 Processor accesses

System register access allows the Cortex®-R82 processor to directly access certain Debug registers through the use of dedicated system instructions (MSR and MRS instructions).

The external debug interface enables debug agents running on a core within the Cortex®-R82 processor to access the Debug registers. Access to the debug registers is partitioned as follows:

Debug registers

This function is System register based. You can access the core Debug registers by using the dedicated system registers. See [13.7 Debug register summary](#) on page 248 for more information on Debug registers.

Performance Monitors registers

This function is System register based. You can access both the core and cluster Performance Monitors registers by using the dedicated system registers. See [14.6 PMU register summary](#) on page 266 for more information on Performance Monitors registers.

Trace registers

This function is System register based. You can access the core trace registers by using the dedicated system registers. See [15.6 ETM register summary](#) on page 276 for more information on trace registers.

13.3.2 Effects of resets on Debug registers

Core and cluster Cold resets and Warm resets that are generated by programming the *Power Policy Units* (PPUs) affect the Debug related registers.

A core Cold reset affects all resettable registers in a specific core including the debug, *Embedded Trace Macrocell* (ETM), and *Reliability, Availability, and Serviceability* (RAS) registers.

A core Warm reset affects all resettable registers in a specific core except the debug, ETM, and RAS registers.

A cluster Cold reset affects all resettable registers in the Cortex®-R82 processor and the DebugBlock except the *Power Policy Unit* (PPU).

A cluster Warm reset affects all resettable registers in the Cortex®-R82 processor except the DebugBlock, PPU, Utility bus, debug, ETM, and RAS registers.

See [6.7 Explicit resetting of cluster and cores and debug recovery](#) on page 88 for more information for reset control with the PPU.

13.3.3 External access permissions to debug registers

External access permission to the Debug registers depends on the conditions at the time of the access.

The following table describes the Cortex®-R82 processor response to accesses through the external debug interface for different access conditions.

Table 13-1: External access conditions to registers

Name	Condition	Effect on access permissions
Off	Core power domain is On as indicated by reading EDPRSR.PU as 1	Access to this field of the EDPRSR register is <i>Read-As-One</i> (RAO) in accordance with Armv8.3 debug over powerdown. When the core power domain is in a powerup state, the debug registers in the core power domain can be accessed. When the core power domain is Off, accesses to the debug registers in the core power domain, including EDPRSR, return an error.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
EDAD	<code>AllowExternalDebugAccess() == FALSE</code>	External debug access is disabled for Non-secure accesses to this register. This causes an error for certain reads and writes to Debug registers through the external debug interface. When an error is returned because of an EDAD condition code, and this is the highest priority error condition, EDPRSR.SDAD is set to 1. Otherwise SDAD is unchanged.
Default	-	None of the conditions apply. This is normal access.

13.3.4 Breakpoints and watchpoints

The Cortex®-R82 processor supports six breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC).

A breakpoint control register (DBGBCR<n>_EL1) and a breakpoint value register (DBGBVR<n>_EL1) forms a breakpoint<n> where <n> is 0-5. These two registers are referred to as a *Breakpoint Register Pair* (BRP).

Four of the breakpoints (BRP 0-3) match only to the virtual address and the other two (BRP 4 and 5) match against either the *Virtual Address* (VA) or context ID, or the *Virtual Machine ID* (VMID). All the watchpoints can be linked to two breakpoints (BRP 4 and 5) to enable a memory request to be trapped in a given process context.

13.4 Debug events

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

The Cortex®-R82 processor 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.

See the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#) for more information about the debug events.

13.4.1 Watchpoint debug events

Watchpoint debug events are always synchronous in the Cortex®-R82 processor.

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.

13.4.2 Debug OS Lock

Debug OS Lock is set by the core Cold reset and cluster Cold reset.

For normal behavior of debug events and Debug register accesses, Debug OS Lock must be cleared either through self-hosted debug performing a System register access or through external debug.

For more information, see the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

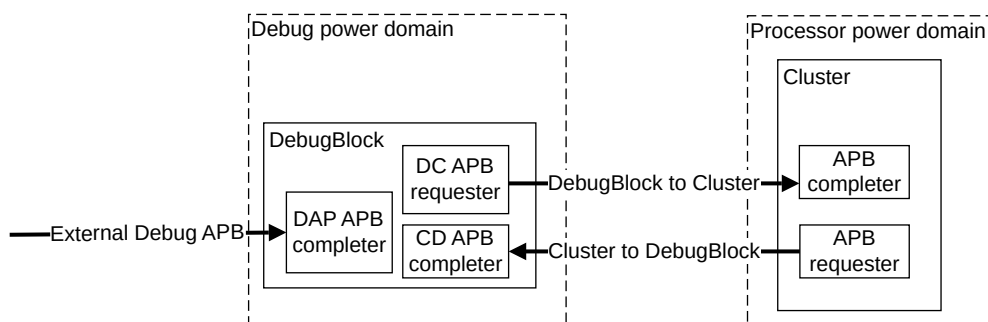
13.5 The DebugBlock

The DebugBlock combines the functions, registers, and interfaces that are required for debug over powerdown.

The DebugBlock is provided as a separate component to allow implementation of it in a separate power domain from the cluster. Having a separate debug power domain allows the connection to a debugger be maintained while the Cortex®-R82 processor is powered down. The Cortex®-R82 processor also allows powering down the DebugBlock when debug is not in process.

The following diagram shows how the DebugBlock is connected to the cluster.

Figure 13-3: Debug APB connections



The DebugBlock has three APB interfaces:

External Debug APB (DAP APB)

An AMBA® APB5 completer interface, allowing communication with an external debugger, for example through a CoreSight™ *Debug Access Port* (DAP).

All debug register read and write requests from an external debugger are received on this bus.

Bus Protection is not provided on this interface since the debug block is expected to be powered down when fault detection is required.

DebugBlock to cluster (DC APB)

An AMBA® APB5 requester interface that is connected to the cluster. It sends all debug register read and write requests to the cluster.

CTI output trigger events are sent to the cluster as trigger event requests on this bus.

Cluster to DebugBlock (CD APB)

An AMBA® APB5 completer interface that is connected to the cluster. It receives CTI input trigger event requests from the cluster.

Debug register reads and writes

The DebugBlock holds all the debug registers that are implemented in the Debug power domain. Registers implemented in the Debug power domain are specified in the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

Accesses through the DAP APB interface to Debug power domain registers are handled internally by the DebugBlock. Accesses through the DAP APB interface to processor power domain (PDCLUSTER or PDCPU<m>) registers are passed on to the cluster through the DC APB interface.

CTI trigger events

Trigger events are transferred between the DebugBlock and cluster through the CD APB and DC APB interfaces.

Input trigger events

Input trigger events are sent from the cluster to the CTIs through the CD APB as write transactions.

Output trigger events

Output trigger events are sent from the CTIs to the cluster through the DC APB multicast trigger requests..

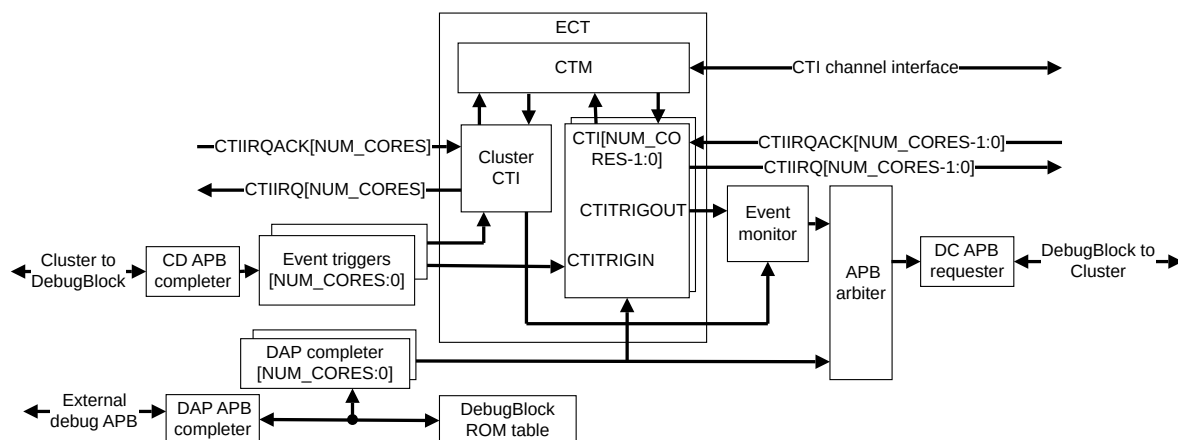
DebugBlock power states

The DebugBlock supports two power modes: ON, and OFF. The DebugBlock sends powerup and powerdown requests to the external power controller on its external Q-Channel interface. In the OFF mode, the DebugBlock does not initiate any APB accesses and all incoming APB accesses to the DebugBlock receive a PSLVERR response.

13.5.1 DebugBlock components

The following figure shows the DebugBlock components.

Figure 13-4: DebugBlock block diagram



The CTIs shown in the diagram includes both the CTIs attached to each of the cores [NUM_CORES-1:0] and the cluster CTI. NUM_CORES has a value of the total number of cores that are implemented.

ECT

The DebugBlock implements the *Embedded Cross Trigger* (ECT).

DebugBlock ROM table

The DebugBlock ROM table holds the address decoding for each debug component in the DebugBlock and one entry pointing to the cluster ROM table. The DebugBlock ROM table complies with the [Arm® CoreSight™ Architecture Specification v3.0](#) and supports the v8 debug address map.

Event monitor

The event monitor converts changes in CTI output triggers to APB write transactions.

Event triggers

The event triggers convert APB write transactions to CTI input triggers.

APB arbiter

The DC APB transfers both register accesses and CTI output trigger events. The APB arbiter multiplexes the two sources of transactions.

DAP completer

The DAP completer holds copies of registers in the debug power domain.

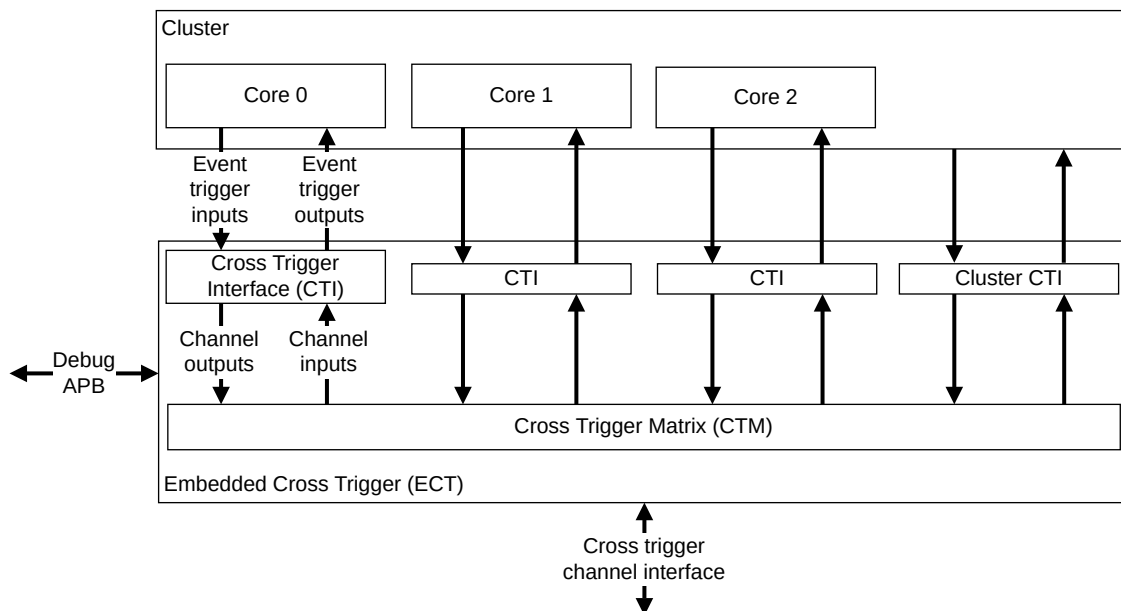
13.6 ECT

The *Embedded Cross Trigger* (ECT) allows debug events to be sent between the cores within the Cortex®-R82 processor.

The ECT provides a *Cross Trigger Interface* (CTI) for each core in the cluster and a CTI at the cluster level. The CTIs are interconnected through a *Cross Trigger Matrix* (CTM) implemented at the DebugBlock to send debug and trace events between the cores.

The following diagram shows a conceptual view of the trigger event inputs and outputs between the cores and ECT.

Figure 13-5: Embedded Cross Trigger concept



The CTIs selectively send trigger events to the CTM on their respective channel outputs. The CTIs receive trigger events from the CTM on their channel inputs.

Trigger events are transferred between CTIs over the channel interface. The CTM connects the channel interface to the channel inputs and channel outputs of the CTIs.

External interfaces

The external cross-trigger channel interface, from the CTM, allows cross-triggering between SoC external devices.

The Debug APB provides access to the CTI registers. This allows an external debugger to configure the trigger event routing, and send events to cores, for example, to put a core into Debug state.

CTI registers

Registers in the CTI:

- Control the mapping of the input trigger events to channel outputs.
- Control the mapping of the channel inputs to output trigger events.
- Capture the state of input and output trigger events.
- Set, clear, or pulse output trigger events.

13.6.1 Supported debug and trace trigger events

The CTIs each have ten input and output trigger events that are mapped onto the debug and trace events in the cores and ELAs.

From the CTI to the core

The debug and trace trigger events from the CTI to the core are:

Debug request trigger event

A trigger event sent from the CTI to the core to force the core into Debug state.

Restart request trigger event

A trigger event sent from the CTI to the core to request the core to exit Debug state.

Generic CTI interrupt trigger event

A trigger event sent from the CTI to the GIC.

ETM trace input trigger events

Four trigger events sent from the CTI to the ETM trace in the core.

ELA input trigger events

Two trigger events sent from the CTI to the ELA attached to the core. If there is no ELA, the ELA trigger events are tied LOW.

From the core to the CTI

The debug and trace events from the core to the CTI are:

Cross-halt trigger event

A trigger event sent from the core to the CTI when the core enters Debug state.

Performance Monitoring Unit (PMU) overflow trigger event

A trigger event sent from the core to the CTI when a PMU counter overflows.

ETM trace output trigger events

Four trigger events sent from the ETM in the core to the CTI.

ELA output trigger events

Two trigger events sent from the ELA (attached to the core) to the CTI.

Similar to core CTIs, the cluster CTI has ten input and output trigger events. However, only the PMU, Generic CTI, and ELA entries are relevant. This is because there is a PMU and an ELA component at the cluster level, but there are no cluster debug and ETM components.

From the cluster CTI to the cluster

The trigger events from the cluster CTI to the cluster are:

Generic CTI interrupt trigger event

A trigger event sent from the cluster CTI to the cluster GIC.

Cluster ELA input trigger events

Two trigger events sent from the cluster CTI to the cluster ELA.

From the cluster to the cluster CTI

The trigger events from the cluster to the cluster CTI are:

PMU overflow trigger event

A trigger event sent from the cluster to the cluster CTI when the cluster PMU counter overflows.

Cluster ELA output trigger events

Two trigger events from the cluster ELA to the cluster CTI.

13.6.2 CTI triggers

The DebugBlock implements a *Cross Trigger Interface* (CTI) per core and a CTI for the cluster. Each CTI has 10 CTI inputs and 10 CTI output triggers. All cores in the Cortex®-R82 processor have the same CTI trigger mapping.

Core CTI input trigger events

The following table shows how events are mapped onto core CTI input triggers.

Table 13-2: Allocation of core CTI trigger inputs

Trigger number	Source	Destination	Type	Event description
0	Core Debug	CTI	Pulse	Cross-halt trigger event
1	Core PMU	CTI	Pulse	Performance Monitoring Unit (PMU) Overflow trigger event
2-3	-	-	-	Reserved
4-7	Core ETM	CTI	Pulse	<i>Embedded Trace Macrocell</i> (ETM) trace external output trigger events
8-9	Core ELA	CTI	Pulse	<i>Embedded Logic Analyzer</i> (ELA) CTTRIGOUT[1:0] trigger events

Core CTI output trigger events

The following table shows how events are mapped onto core CTI output triggers.

Table 13-3: Allocation of core CTI trigger outputs

Trigger number	Source	Destination	Type	Event description
0	CTI	Core Debug	Level	Debug Request trigger event
1	CTI	Core Debug	Pulse	Restart Request trigger event
2	CTI	GIC	Pulse	Generic CTI Interrupt trigger event
3	-	-	-	Reserved

Trigger number	Source	Destination	Type	Event description
4-7	CTI	ETM	Pulse	ETM trace external input trigger events
8-9	CTI	ELA	Pulse	ELA CTTRIGIN[1:0] trigger events

Cluster CTI input trigger events

The following table shows how events are mapped onto the cluster CTI input triggers.

Table 13-4: Allocation of cluster CTI trigger inputs

Trigger number	Source	Destination	Type	Event description
0	-	-	-	Reserved
1	Cluster PMU	CTI	Pulse	PMU Overflow trigger event
2-7	-	-	-	Reserved
8-9	Cluster ELA	CTI	Pulse	Cluster ELA CTTRIGOUT[1:0]

Cluster CTI output trigger events

The following table shows how events are mapped onto the cluster CTI output triggers.

Table 13-5: Allocation of cluster CTI trigger outputs

Trigger number	Source	Destination	Type	Event description
0-1	-	-	-	Reserved
2	CTI	GIC	Pulse	Generic CTI Interrupt trigger event
3-7	-	-	-	Reserved
8-9	CTI	Cluster ELA	Pulse	Cluster ELA CTTRIGIN[1:0]

13.6.3 CTI register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary table for the Cortex®-R82 processor *Cross Trigger Interface* (CTI) registers [B.2.1.6 External CTI registers summary](#) on page 1608.

13.7 Debug register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary table for the Cortex®-R82 processor AArch64 Debug registers in [A.1.5 AArch64 Debug registers summary](#) on page 294.

You can find the register summary table for the Cortex®-R82 processor external Debug registers in [B.2.1.1 External Debug registers summary](#) on page 1601.

14. PMU

This chapter describes the *Performance Monitoring Unit* (PMU).

14.1 About the PMU

The Cortex®-R82 processor includes *Performance Monitoring Units* (PMUs) to assist software profiling and performance debugging. The PMUs implement the PMUv3 architecture and include Arm®v8.4 PMU extensions.

The PMUs enable you to gather various statistics on the operation of each core and the cluster and their memory system during runtime.

The Cortex®-R82 processor implements:

- One PMU per core to monitor events local to the core such as L1 cache linefills.
- One PMU at the cluster level to monitor events that are shared by the cores such as L2 cache linefills or *LLRAM Coherency Unit* (LCU) accesses.

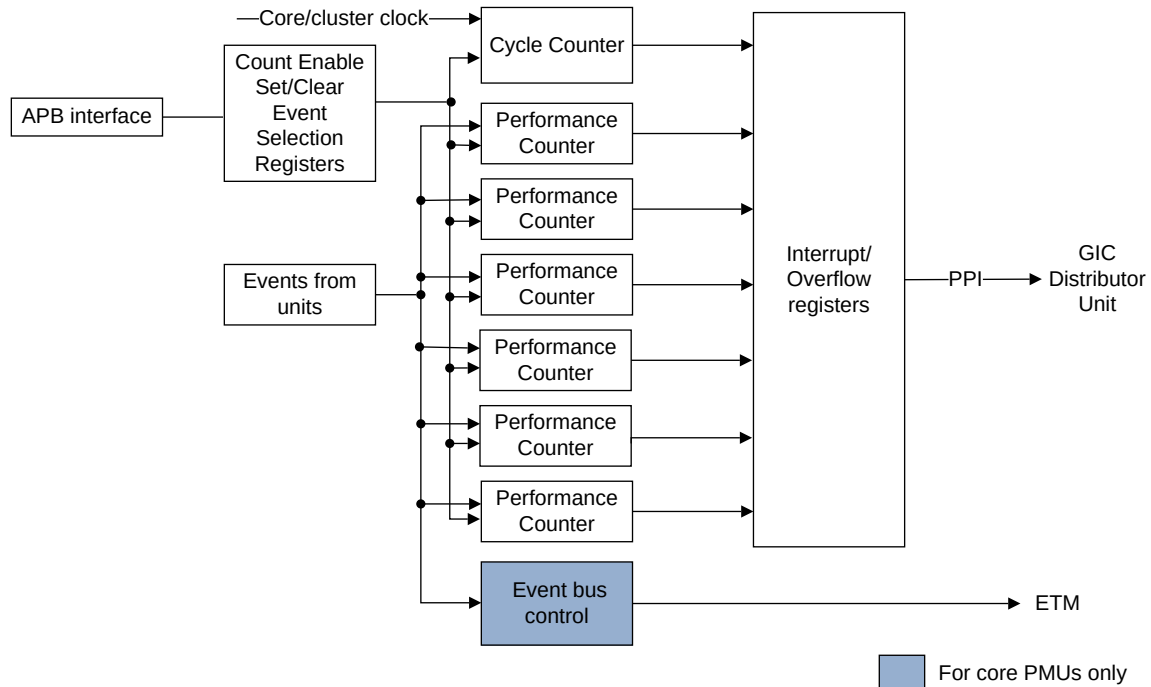
Some of the events from the per-core PMUs are also exported for use by the respective Trace units.

Each PMU implements six 64-bit event counters. Each counter in core PMUs can count any of the events available in that core. Each counter in the cluster PMU can count any of the events available in the cluster. The absolute counts that are recorded might vary because of pipeline effects. This has negligible effect except in cases where the counters are enabled for a very short time.

Software running on the cores in the Cortex®-R82 processor can access the PMUs through System registers. Each core is able to access its own private PMU and the shared cluster PMU. All the core PMUs and the cluster PMU are accessible to an external agent, such as an external debug host, through the Debug APB bus.

The cluster PMU is not accessible when the cluster is in Warm reset, such as during the OFF_EMU power mode.

The following figure shows the major blocks inside the PMU.

Figure 14-1: PMU block diagram

14.2 PMU functional description

This section describes the functionality of the PMU.

Event interface

Events from units from across the design are provided to the PMU.

System registers and APB interface

You can program the core and cluster PMU registers using the System registers or the external APB interface.

Counters

The PMU has 32-bit event counters that increment when they are enabled, based on events, and a 64-bit cycle counter.

14.3 External register access permissions to the PMU registers

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

The following table describes the core response to accesses through the external debug and memory-mapped interfaces.

Table 14-1: External register conditions

Name	Condition	Description
Off	EDPRSR.PU is 0	Access to this field is Read-As-One (RAO) in accordance with Armv8.3 debug over powerdown. When the core power domain is in a powerup state, the PMU registers in the core power domain can be accessed. When the core power domain is off, accesses to the PMU registers in the core power domain, including debug registers and EDPRSR, return an error.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
Default	-	None of the conditions apply, normal access.

14.4 PMU events

This section describes the core and cluster *Performance Monitoring Unit* (PMU) events of the Cortex®-R82 processor.

14.4.1 Core PMU events

Performance Monitoring Units (PMUs) private to each core collect events from the units within the core and use event codes to reference these events.

Core PMU events are architecturally, microarchitecturally, or implementation defined. Architectural, microarchitectural, and implementation defined core PMU events are the same for each core within the Cortex®-R82 processor.

14.4.1.1 Architectural core PMU events

The following table lists the architectural events in each core.



Some events are exported to the *Embedded Trace Macrocell* (ETM). The ETM can directly select the events based on their *Performance Monitoring Unit* (PMU) event code. You can use these events to build complex triggers.

Table 14-2: Architectural core PMU events

Event code	ETM code	Mnemonic	Description
0x0000	0x-	SW_INCR	Instruction architecturally executed, condition code check pass, software increment.
0x0006	0x09	LD_RETIRED	Instruction architecturally executed, condition code check pass, load. This counts all load and prefetch instructions. This includes the v8.1 atomic instructions, other than the ST* variants.
0x0007	0x0A	ST_RETIRED	Instruction architecturally executed, condition code check pass, store. This counts all store instructions, and DC ZVA. This includes all the v8.1 atomic instructions. Store-exclusive instructions which fail are not counted.
0x0008	0x0B	INST_RETIRED	Instruction architecturally executed. This counts all retired instructions, including those that fail their condition code check.
0x0009	0x0C	EXC_TAKEN	Exception taken
0x000A	0x0D	EXC_RETURN	Instruction architecturally executed, condition code check pass, exception return
0x000B	0x0E	CID_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, write to CONTEXTIDR_EL1. Writes to CONTEXTIDR_EL12 are not counted.
0x000C	0x0F	PC_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, software change of the PC.
0x000D	0x10	BR_IMMED_RETIRED	Instruction architecturally executed, immediate branch. ISBs are not counted.
0x000E	0x11	BR_RETURN_RETIRED	Instruction architecturally executed, condition code check pass, procedure return
0x000F	0x12	UNALIGNED_LDST_RETIRED	Instruction architecturally executed, condition code check pass, unaligned load or store
0x001C	0x1E	TTBR_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, write to TTBRx_EL1.
0x001E	0x-	CHAIN	Odd performance counter chain mode
0x0021	0x21	BR_RETIRED	Instruction architecturally executed, branch. This includes any instruction that is in the branch pipeline.

14.4.1.2 Microarchitectural core PMU events

The following table lists the microarchitectural events in each core.

Table 14-3: Microarchitectural core PMU events

Event code	ETM code	Mnemonic	Description
0x0001	0x04	L1I_CACHE_REFILL	L1 instruction cache refill. Counts any instruction fetch which misses in the cache and starts a new cache refill. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted. This event counts the sum of MM_L1I_CACHE_REFILL and LLRAM_L1I_CACHE_REFILL.
0x0002	0x05	L1I_TLB_REFILL	L1 instruction TLB refill. Counts any refill of the instruction L1-MMS from the L2 MMS. This includes refills which result in a translation fault. This includes accesses to either the MPUs or the TLB. TLB maintenance instructions are not counted. This event counts regardless of whether translation is enabled.
0x0003	0x06	L1D_CACHE_REFILL	L1 data cache refill. Counts any load or store operation or pagewalk access which causes data to be read from outside the L1, including accesses which do not allocate into L1. Cache maintenance instructions and prefetches are not counted. Stores of an entire cache line are not counted, even if they make a coherency request outside the L1. Partial cache line writes which do not allocate into the L1 cache are not counted. Non-cacheable accesses are not counted. This event counts the sum of L1D_CACHE_REFILL_RD and L1D_CACHE_REFILL_WR.
0x0004	0x07	L1D_CACHE	L1 data cache access. Counts any load or store operation or pagewalk access which looks up in the L1 data cache. In particular, any access which could increment the L1D_CACHE_REFILL event causes this event to count. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted. This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.
0x0005	0x08	L1D_TLB_REFILL	L1 data TLB refill. Counts any refill of the data L1-MMS from the L2 MMS. This includes refills which result in a translation fault. TLB maintenance instructions are not counted. This event counts regardless of whether translation is enabled.
0x0010	0x13	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed. This counts any predictable branch instruction (in other words, any type of instruction that the IFU can predict) which is mispredicted (either due to dynamic misprediction, or because the MMU is off and the branches are statically predicted not taken).
0x0011	0x-	CPU_CYCLES	The number of core clock cycles. Counts even when the core is in WFI or WFE state.
0x0012	0x14	BR_PRED	Predictable branch speculatively executed. Counts all predictable branches (superset of BR_MIS_PRED).
0x0013	0x15	MEM_ACCESS	Data memory access. Counts memory accesses due to load or store instructions. Does not count instruction fetches, cache maintenance instructions, translation table walks or prefetches. This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.
0x0014	0x0x16	L1I_CACHE	L1 instruction cache access. Counts any instruction fetch which accesses the L1 I-cache. Cache maintenance instructions are not counted. Non-cacheable accesses are not counted. This event is a sum of MM_L1I_CACHE and LLRAM_L1I_CACHE.
0x0015	0x17	L1D_CACHE_WB	L1 data cache write-back. Counts any write back of data from the L1 data cache to L2. This counts both victim line evictions and snoops, including cache maintenance operations. Invalidations which do not result in data being transferred out of the L1 are not counted. Does not count any full-line writes which write to L2 without writing L1 (for example, write-streaming mode).
0x0019	0x1B	BUS_ACCESS	Bus access. Counts for every beat of data transferred over the data channels between the core and the SCU. If both read and write data beats are transferred on a given cycle, this event is counted twice on that cycle. This event counts the sum of BUS_ACCESS_RD and BUS_ACCESS_WR.

Event code	ETM code	Mnemonic	Description
0x001A	0x1C	MEMORY_ERROR	Local memory error. Counts any Correctable or Uncorrectable memory error (ECC or parity) in the protected CPU RAMs. Counts only when error recording is enabled.
0x001B	0x1D	INST_SPEC	Operation speculatively executed. This event counts every instruction issued from the Iss pipeline stage.
0x001D	0x--	BUS_CYCLES	Bus cycles. The event duplicates CPU_CYCLES.
0x001F	0x1F	L1D_CACHE_ALLOCATE	L1 data cache allocation without refill. The counter increments on every attributable write that writes an entire line into the L1 cache without fetching from outside the L1 cache, for example: Writes merged to a full cache line in the store buffer or a DC ZVA operation.
0x0020	0x20	L2D_CACHE_ALLOCATE	L2 unified cache allocation without refill. The counter increments on every attributable write that writes an entire line into the L2 cache without fetching from outside the L1 or L2 cache, for example: Writes merged to a full cache line in the store buffer or a write-back from L1 to L2 cache or a DC ZVA operation.
0x0022	0x22	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch. Counts any branch counted by BR_RETIRED which is not correctly predicted and causes a pipeline flush.
0x0023	0x23	STALL_FRONTEND	No operation issued due to the frontend. Counts on any cycle when no operations are issued due to the instruction queue being empty. This event is a sum of STALL_FRONTEND_CACHE and STALL_FRONTEND_TLB.
0x0024	0x24	STALL_BACKEND	No operation issued due to backend. Counts on any cycle when no operations are issued due to a pipeline stall. This event is a sum of all STALL_BACKEND_* events.
0x0025	0x25	L1D_TLB	L1 data TLB access. Counts any load or store operation which accesses the data L1-MMS. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether translation is enabled.
0x0026	0x26	L1I_TLB	L1 instruction TLB access. Counts any instruction fetch which accesses the instruction L1-MMS. This event counts regardless of whether translation is enabled.
0x002D	0x2A	L2D_TLB_REFILL	Attributable L2 unified TLB refill. Counts on any refill of the L2 TLB, caused by either an instruction or data access. This event does not count if the MMU is disabled. This event counts only when the MMU is present.
0x002F	0x2B	L2D_TLB	Attributable L2 unified TLB access. Counts on any access to the L2 TLB (caused by a refill of any of the L1 TLBs). This event does not count if the MMU is disabled. This event counts only when the MMU is present.
0x0036	0x2C	LL_CACHE_RD	Last Level cache access, read, main manager address. If IMP_CPUECTLR_EL1.EXTLLC is set: Counts SYS_CACHE_RD. If L2 cache is present counts L2_CACHE_RD otherwise counts L1_CACHE_RD.
0x0037	0x2D	LL_CACHE_MISS_RD	Last Level cache miss, read, main manager address. If IMP_CPUECTLR_EL1.EXTLLC is set: counts SYS_CACHE_MISS_RD. If L2 cache is present counts SYS_CACHE_RD otherwise counts L1D_CACHE_REFILL_RD.
0x0038	0x19	REMOTE_ACCESS_RD	Access to another socket in a multi-socket system, read. Counts any read transaction which returns a data source of remote.
0x0039	0x--	L1D_CACHE_LMISS_RD	The counter counts each access counted by L1D_CACHE that incurs additional latency because it returns data from outside the L1 data cache. This counter does not count: A miss that does not cause a new cache refill but is satisfied from a previous miss.
0x003A	0x--	OP_RETIRED	The counter counts each operation counted by OP_SPEC that would be executed in a simple sequential execution of the program.
0x003B	0x--	OP_SPEC	The counter counts the number of operations executed by the PE, including those that are executed speculatively and would not be executed in a simple sequential execution of the program.

Event code	ETM code	Mnemonic	Description
0x003C	0x-	STALL	The counter counts every Attributable cycle on which no Attributable instruction or operation was sent for execution on this PE. This event is the union of STALL_FRONTEND and STALL_BACKEND.
0x003D	0x-	STALL_SLOT_BACKEND	Counts each slot counted by STALL_SLOT where no attributable instruction or operation was sent for execution because the backend is unable to accept one of: the instruction operation available for the PE on the slot or any operations on the slot.
0x003E	0x-	STALL_SLOT_FRONTEND	Counts each slot counted by STALL_SLOT where no attributable instruction or operation was sent for execution because there was no attributable instruction or operation available to issue from the PE from the frontend for the slot.
0x003F	0x-	STALL_SLOT	The counter counts on each attributable cycle the number of instruction or operation slots that were not occupied by an instruction or operation attributable to the PE.
0x4006	0x-	L1I_CACHE_LMISS	The counter counts each access counted by L1I_CACHE that incurs additional latency because it returns instructions from outside the L1 instruction cache. This counter does not count: A miss that does not cause a new cache refill but is satisfied from a previous miss.
0x4020	0x-	LDST_ALIGN_LAT	The counter counts each access counted by MEM_ACCESS that, due to the alignment of the address and size of data being accessed, incurred additional latency.
0x4021	0x-	LD_ALIGN_LAT	The counter counts each memory-read access counted by LDST_ALIGN_LAT.
0x4022	0x-	ST_ALIGN_LAT	The counter counts each memory-write access counted by LDST_ALIGN_LAT.

14.4.1.3 Implementation defined core PMU events

The following table lists the implementation defined core events and the numbers that the *Performance Monitoring Unit* (PMU) in that core uses to reference the events.

Table 14-4: implementation defined core PMU events

Event code	ETM code	Mnemonic	Description
0x00C1	0x-	L2D_CACHE_REFILL_PREFETCH	A stash request to prefetch a line into the L2 cache initiated from the core. This includes stash requests to lines that are already in the L2 cache. If the cluster does not contain an L2 cache, this event does not count as the prefetcher is implicitly disabled.
0x00C2	0x-	L1D_CACHE_REFILL_PREFETCH	L1 data cache refill due to prefetch. Counts any linefills from the prefetcher which cause an allocation into the L1 D-cache.
0x00C3	0x-	L2D_WS_MODE	L2 cache write streaming mode. Counts for each cycle where the core is in write-streaming mode and not allocating writes into the L2 cache.
0x00C4	0x-	L1D_WS_MODE_ENTRY	L1 data cache entering write streaming mode. Counts for each entry into write-streaming mode.
0x00C5	0x-	L1D_WS_MODE	L1 data cache write streaming mode. Counts for each cycle where the core is in write-streaming mode and not allocating writes into the L1 D-cache.
0x00C9	0x-	BR_COND_PRED	Predicted conditional branch executed. Counts when any branch which can be predicted by the conditional predictor is retired. This event still counts when branch prediction is disabled due to the MMU being off.

Event code	ETM code	Mnemonic	Description
0x00CA	0x-	BR_INDIRECT_MIS_PRED	Indirect branch mis-predicted. Counts when any indirect branch which can be predicted by the BTAC is retired, and has mis-predicted for either the condition or the address. This event still counts when branch prediction is disabled due to the MMU being off.
0x00CB	0x-	BR_INDIRECT_ADDR_MIS_PRED	Indirect branch mis-predicted due to address mis-compare. Counts when any indirect branch which can be predicted by the BTAC is retired, was taken and correctly predicted the condition, and has mis-predicted the address. This event still counts when branch prediction is disabled due to the MMU being off.
0x00CC	0x-	BR_COND_MIS_PRED	Conditional branch mis-predicted. Counts when any branch which can be predicted by the conditional predictor is retired, and has mis-predicted the condition. This event still counts when branch prediction is disabled due to the MMU being off. Conditional indirect branches which correctly predicted the condition but mis-predicted on the address do not count this event.
0x00CD	0x-	BR_INDIRECT_ADDR_PRED	Indirect branch with predicted address executed. Counts when any indirect branch which can be predicted by the BTAC is retired, was taken and correctly predicted the condition. This event still counts when branch prediction is disabled due to the MMU being off.
0x00CE	0x-	BR_RETURN_ADDR_PRED	Procedure return with predicted address executed. Counts when any procedure return which can be predicted by the CRS is retired, was taken and correctly predicted the condition. This event still counts when branch prediction is disabled due to the MMU being off.
0x00CF	0x-	BR_RETURN_ADDR_MIS_PRED	Procedure return mis-predicted due to address mis-compare. Counts when any procedure return which can be predicted by the CRS is retired, was taken and correctly predicted the condition, and has mis-predicted the address. This event still counts when branch prediction is disabled due to the MMU being off.
0x00D0	0x-	L2D_LLWALK_TLB	L2 TLB last-level walk cache access. This event does not count if the MMU is disabled. This event counts only when the MMU is present.
0x00D1	0x-	L2D_LLWALK_TLB_REFILL	L2 TLB last-level walk cache refill. This event does not count if the MMU is disabled. This event counts only when the MMU is present.
0x00D2	0x-	L2D_L2WALK_TLB	L2 TLB level-2 walk cache access. This event count accesses to the level-2 walk cache where the last-level walk cache has missed. The event only counts when the translation regime of the pagewalk uses level 2 descriptors. This event does not count if the MMU is disabled. This event counts only when the MMU is present.
0x00D3	0x-	L2D_L2WALK_TLB_REFILL	L2 TLB level-2 walk cache refill. This event does not count if the MMU is disabled. This event counts only when the MMU is present.
0x00E1	0x-	STALL_FRONTEND_CACHE	No operation issued due to the frontend, cache miss. Counts every cycle the DPU IQ is empty and there is an instruction cache miss being processed
0x00E2	0x-	STALL_FRONTEND_TLB	No operation issued due to the frontend, TLB miss. Counts every cycle the DPU IQ is empty and there is an instruction L1-TLB miss being processed
0x00E4	0x-	STALL_BACKEND_ILOCK	No operation issued due to the backend, interlock. Counts every cycle that issue is stalled due to a dependency. Stall cycles due to a stall in Wr (typically awaiting load data) are excluded.
0x00E5	0x-	STALL_BACKEND_ILOCK_AGU	No operation issued due to the backend, interlock, AGU. Counts every cycle that issue is stalled due to a load/store instruction waiting for data to calculate the address in the AGU. Stall cycles due to a stall in Wr (typically awaiting load data) are excluded.

Event code	ETM code	Mnemonic	Description
0x00E6	0x-	STALL_BACKEND_ILOCK_FPU	No operation issued due to the backend, interlock, FPU. Counts every cycle that issue is stalled due to a dependency of an FPU/NEON instruction. Stall cycles due to a stall in the Wr stage (typically awaiting load data) are excluded.
0x00E7	0x-	STALL_BACKEND_LD	No operation issued due to the backend, load. Counts every cycle there is a stall in the Wr stage due to a load.
0x00E8	0x-	STALL_BACKEND_ST	No operation issued due to the backend, store. Counts every cycle there is a stall in the Wr stage due to a store.
0x00E9	0x-	STALL_BACKEND_LD_CACHE	No operation issued due to the backend, load, cache miss. Counts every cycle there is a stall in the Wr stage due to a load which is waiting on data (due to missing the cache or being non-cacheable).
0x00EA	0x-	STALL_BACKEND_LD_TLB	No operation issued due to the backend, load, TLB miss. Counts every cycle there is a stall in the Wr stage due to a load which has missed in the L1 TLB.
0x00EB	0x-	STALL_BACKEND_ST_STB	No operation issued due to the backend, store, STB full. Counts every cycle there is a stall in the Wr stage due to a store which is waiting due to the STB being full.
0x00EC	0x-	STALL_BACKEND_ST_TLB	No operation issued due to the backend, store, TLB miss. Counts every cycle there is a stall in the Wr stage due to a store which has missed in the L1 TLB.
0x00ED	0x-	STALL_BACKEND_LD_RAW	No operation issued due to the backend, load, stalled due to a read-after-write hazard.
0x0300	0x-	BR_NANO_IMM_ACCESS	Fetch pipeline accessed nano predictors for an immediate branch.
0x0301	0x-	BR_NANO_IMM_HIT	An immediate branch hit in the nano predictors in the fetch pipeline.
0x0302	0x-	BR_NANO_IMM_MIS_PRED	An immediate branch mis-predicted in the nano predictors in the fetch pipeline.
0x0304	0x-	BR_NANO_COND_ACCESS	Fetch pipeline accessed nano predictors for a conditional branch.
0x0305	0x-	BR_NANO_COND_HIT	A conditional branch hit in the nano predictors in the fetch pipeline.
0x0306	0x-	BR_NANO_COND_MIS_PRED	A conditional branch mis-predicted in the nano predictors in the fetch pipeline.
0x0307	0x-	BR_NANO_INDIRECT_ACCESS	Fetch pipeline accessed nano predictors for an indirect branch.
0x0308	0x-	BR_NANO_INDIRECT_HIT	An indirect branch hit in the nano predictors in the fetch pipeline.
0x0309	0x-	BR_NANO_INDIRECT_MIS_PRED	An indirect branch mis-predicted in the nano predictors in the fetch pipeline.
0x030a	0x-	BR_NANO_RETURN_ACCESS	Fetch pipeline accessed nano predictors for a return instruction.
0x030b	0x-	BR_NANO_RETURN_HIT	A return instruction hit in the nano predictors in the fetch pipeline.
0x030c	0x-	BR_NANO_RETURN_MIS_PRED	A return instruction mis-predicted in the nano predictors in the fetch pipeline.
0x030d	0x-	MM_L1I_PREFETCH_ACCESS	This event counts accesses to L1 Icache initiated by the prefetcher to an address mapped to the main manager port.
0x030e	0x-	MM_L1I_PREFETCH_REFILL	This event counts Icache refills initiated by the prefetcher to an address mapped to the main manager port.
0x030f	0x-	LLRAM_L1I_PREFETCH_ACCESS	This event counts accesses to L1 Icache initiated by the prefetcher to an address mapped to the LLRAM port.
0x0310	0x-	LLRAM_L1I_PREFETCH_REFILL	This event counts Icache refills initiated by the prefetcher to an address mapped to the LLRAM port.
0x0320	0x-	SYS_CACHE_RD	This event counts any cacheable read transaction which returns a data source of interconnect cache or inter-cluster peer.

Event code	ETM code	Mnemonic	Description
0x0321	0x-	SYS_CACHE_MISS_RD	This event counts any cacheable read transaction which returns a data source of DRAM.
0x0322	0x-	LLPP_ACCESS_RD	LLPP access, read, counts for every unique read request sent to the LLPP read address channel.
0x0323	0x-	LLPP_ACCESS_WR	LLPP access, write, counts for every unique write request sent to the LLPP write address channel.
0x0324	0x-	LLPP_ACCESS	Counts accesses made on the LLPP. This event is a sum of LLPP_ACCESS_WR and LLPP_ACCESS_RD.
0x0325	0x-	MM_ACCESS_RD	Bus access, read, Counts for every beat of data transferred over the read data channel between the shared L2 and the core.
0x0326	0x-	MM_ACCESS_WR	Bus access, write. Counts for every beat of data transferred over the write data channel between the core and the L2.
0x0327	0x-	MM_ACCESS	Counts accesses made on the main manager channel between the core and the L2. This event is a sum of MM_ACCESS_WR and MM_ACCESS_RD.
0x0328	0x-	LLRAM_ACCESS_RD	Bus access, read, Counts for every beat of data transferred over the read data channel between the LLRAM coherency unit and the core targetting the LLRAM port.
0x0329	0x-	LLRAM_ACCESS_WR	Bus access, write. Counts for every beat of data transferred over the write data channel between the core and LLRAM coherency unit targetting the LLRAM port.
0x032a	0x-	LLRAM_ACCESS	Counts accesses made between the core and the LLRAM coherency unit targetting the LLRAM port. This event is a sum of LLRAM_ACCESS_WR and LLRAM_ACCESS_RD.
0x032b	0x-	SPP_ACCESS_RD	Bus access, read, Counts for every beat of data transferred over the read data channel between the LLRAM coherency unit and the core targetting the SPP port.
0x032c	0x-	SPP_ACCESS_WR	Bus access, write. Counts for every beat of data transferred over the write data channel between the core and LLRAM coherency unit targetting the SPP port.
0x032d	0x-	SPP_ACCESS	Counts accesses made between the core and the LLRAM coherency unit targetting the SPP port. This event is a sum of SPP_ACCESS_WR and SPP_ACCESS_RD.
0x032e	0x-	LLPP_CYCLES	Low latency peripheral port bus has outstanding transactions (counted on SCLK instead of bus clock)
0x032f	0x-	LLPP_ACTIVE	Low latency peripheral port cycles
0x0330	0x-	TCMS_ACCESS_RD	ACELS access to TCMS, read, counts for every unique read request sent to the TCMS from the ACELS port to this core.
0x0331	0x-	TCMS_ACCESS_WR	ACELS access to TCMS, write, counts for every unique write request sent to the TCMS from the ACELS port to this core.
0x0332	0x1A	TCMS_ACCESS	Counts accesses made between the ACELS port and the core. This event is a sum of TCMS_ACCESS_WR and TCMS_ACCESS_RD.
0x0333	0x18	TCMS_CONTENTION	Counts every stall cycle due to contention of accessing ITCM and DTCM via the ACELS port at the same time.
0x0334	0x-	MM_SNP_ACCESS	Counts every unique snoop request received by the core from the L2. This includes snoops received from the own core, another core or from outside the cluster in the CHI configuration.

Event code	ETM code	Mnemonic	Description
0x0335	0x-	LLRAM_SNP_ACCESS	Counts every unique snoop request received by the core from the LLRAM coherency unit. This includes snoops received from the own core or another core.
0x0336	0x-	SNP_ACCESS	Counts every unique snoop request received by the core. This event is a sum of MM_SNP_ACCESS and LLRAM_SNP_ACCESS.
0x0337	0x-	MM_SNP_ACCESS_EVICT	Counts every unique snoop request received by the core from the L2 that evicts data. This includes snoops received from the own core, another core or from outside the cluster in the CHI configuration.
0x0338	0x-	LLRAM_SILENT_EVICT	Counts every instance of silently evicting a valid LLRAM line in the L1D cache in order to allocate a main manager address in the same set/way.
0x0339	0x-	TCMS_SERIALISATION_CONTENTION	Counts every stall cycle due to contention of read and write channel requesting serialization at the same time.
0x0340	0x-	LLRAM_SNP_ACCESS_EVICT	Counts every unique snoop request received by the core from the LCU that evicts data. This includes only snoops received from the own core.
0x0350	0x-	MM_L1I_CACHE_REFILL	L1 instruction cache refill for an address to the Main Manager (MM) region. Counts any instruction fetch which misses in the cache and starts a new cache refill. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted.
0x0351	0x-	MM_L1I_CACHE	L1 instruction cache access targeting the main manager port.
0x0352	0x-	MM_L1D_CACHE_REFILL_RD	L1 data cache refill, read from the main manager port.
0x0353	0x-	MM_L1D_CACHE_REFILL_WR	L1 data cache refill, write from the main manager port.
0x0354	0x-	MM_L1D_CACHE_REFILL	L1 data cache refill for the main manager port. This event counts the sum of MM_L1D_CACHE_REFILL_RD and MM_L1D_CACHE_REFILL_WR.
0x0355	0x-	MM_L1D_CACHE_RD	L1 data cache access, read targeting the main manager address region.
0x0356	0x-	MM_L1D_CACHE_WR	L1 data cache access, write targeting the main manager address region.
0x0357	0x-	MM_L1D_CACHE	L1 data cache access targeting the main manager port. This event counts the sum of MM_L1D_CACHE_RD and MM_L1D_CACHE_WR.
0x0358	0x-	LLRAM_L1I_CACHE_REFILL	L1 instruction cache refill for an address to the Low-latency RAM (LLRAM) region. Counts any instruction fetch which misses in the cache and starts a new cache refill. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted.
0x0359	0x-	LLRAM_L1I_CACHE	L1 instruction cache access targeting the Low-latency RAM (LLRAM) region.
0x0360	0x-	LLRAM_L1D_CACHE_REFILL_RD	L1 data cache refill, read from the LLRAM port.
0x0361	0x-	LLRAM_L1D_CACHE_REFILL_WR	L1 data cache refill, write from the LLRAM port.
0x0362	0x-	LLRAM_L1D_CACHE_REFILL	L1 data cache refill from the LLRAM port. This event counts the sum of LLRAM_L1D_CACHE_REFILL_RD and LLRAM_L1D_CACHE_REFILL_WR.
0x0363	0x-	LLRAM_L1D_CACHE_RD	L1 data cache access, read targeting the LLRAM address region.
0x0364	0x-	LLRAM_L1D_CACHE_WR	L1 data cache access, write targeting the LLRAM address region.
0x0365	0x-	LLRAM_L1D_CACHE	L1 data cache access targeting the LLRAM address region. This event counts the sum of LLRAM_L1D_CACHE_RD and LLRAM_L1D_CACHE_WR.
0x0366	0x-	LLRAM_L1D_CACHE_REFILL_MERGED	L1 data cache refill from the LLRAM port merged with a previous ongoing refill to the same cacheline address.
0x0367	0x-	LLRAM_L1D_CACHE_REFILL_FAIL	L1 data cache refill from the LLRAM port failed to allocate into the cache.

Event code	ETM code	Mnemonic	Description
0x0368	0x-	LLRAM_L1D_CACHE_REFILL_MM_EVICT	L1 data cache refill from the LLRAM port first requires a main manager line to be evicted from the cache. This event counts the request sent to L2 to evict the MM line from L1D cache.
0x0369	0x-	MM_L1D_CACHE_REFILL_MERGED	L1 data cache refill from the main manager port merged with a previous ongoing refill to the same cacheline address.
0x0370	0x-	TCM_ACCESS_D_RD	I or D TCM accessed from the data side for a read operation.
0x0371	0x-	TCM_ACCESS_D_WR	I or D TCM accessed from the data side for a write operation.
0x0372	0x-	TCM_ACCESS_I	ITCM accessed from the instruction pipe.
0x0373	0x-	ROUTER_STALL	More than one router packet needs arbitrating introducing contention.
0x0374	0x-	MM_STB_FULL	Counts every time the main manager STB slots are full.
0x0375	0x-	LLRAM_STB_FULL	Counts every time the LLRAM STB slots are full. This event also covers accesses to the SPP region.
0x0376	0x-	LLPP_STB_FULL	Counts every time the LLPP STB slots are full.
0x0377	0x-	TCMS_STB_FULL	Counts every time the TC STB slots are full.
0x0378	0x-	BARRIER_STB_FULL	Counts every time the barrier STB slots are full.
0x0379	0x-	L1I_WT_HIT	L1 Instruction cache way tracker hit
0x037a	0x-	L1D_WT_HIT_RD	L1 Data cache way tracker hit for a read operation from the data cache.
0x037b	0x-	L1D_WT_HIT_WR	A lookup into the TLAC hit.
0x0390	0x29	VSCTLR_WR_RETIRED	Instruction architecturally executed, condition code check pass, write to VSCTLR_EL2.
0x0391	0x28	DFB_RETIRED	Instruction architecturally executed, condition code check pass for data full barrier instruction.
0x0392	0x27	EL2_ENTERED	Exception taken to EL2 (hyp mode entry), excluding reset
0x0399	0x-	L1D_TLB_REFILL_PREFETCHER	L1 TLB refill used by the data side prefetcher. Counts any refill of the data L1-MMS from the L2 MMS. This includes refills which result in a translation fault. This event counts regardless of whether translation is enabled.
0x039a	0x-	L1D_TLB_PREFETCHER	L1 TLB access by the data side prefetcher.

14.4.1.4 Recommended implementation defined core PMU events

Arm recommends some implementation defined core PMU events.

The following table shows the recommended implementation defined events that are generated at the core and the numbers that the *Performance Monitoring Unit* (PMU) in that core uses to reference the events.

Table 14-5: Recommended implementation defined core PMU events

Event code	ETM code	Mnemonic	Description
0x0040	0x-	L1D_CACHE_RD	L1 data cache access, read. Counts any load operation or pagewalk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted.

Event code	ETM code	Mnemonic	Description
0x0041	0x-	L1D_CACHE_WR	L1 data cache access, write. Counts any store operation which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_WR event causes this event to count. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted. This event is a sum of MM_L1D_CACHE_WR and LLRAM_L1D_CACHE_WR.
0x0042	0x-	L1D_CACHE_REFILL_RD	L1 data cache refill, read. Counts any load operation or pagewalk access which causes data to be read from outside the L1, including accesses which do not allocate into L1. Cache maintenance instructions and prefetches are not counted. Non-cacheable accesses are not counted. This event is a sum of MM_L1D_CACHE_REFILL_RD and LLRAM_L1D_CACHE_REFILL_RD.
0x0043	0x-	L1D_CACHE_REFILL_WR	L1 data cache refill, write. Counts any store operation which causes data to be read from outside the L1, including accesses which do not allocate into L1. Cache maintenance instructions and prefetches are not counted. Stores of an entire cache line are not counted, even if they make a coherency request outside the L1. Partial cache line writes which do not allocate into the L1 cache are not counted. Non-cacheable accesses are not counted. This event is a sum of MM_L1D_CACHE_REFILL_WR and LLRAM_L1D_CACHE_REFILL_WR.
0x0044	0x-	L1D_CACHE_REFILL_INNER	L1 data cache refill, inner main manager address. Counts any L1 D-cache linefill (as counted by L1D_CACHE_REFILL) which hits in the L2 cache, or another core in the cluster.
0x0045	0x-	L1D_CACHE_REFILL_OUTER	L1 data cache refill, outer main manager address. Counts any L1 D-cache linefill (as counted by L1D_CACHE_REFILL) which does not hit in the L2 cache, or another core in the cluster, and instead obtains data from outside the cluster.
0x0050	0x-	L2D_CACHE_RD	L2 cache access, read, main manager address. This event counts any cacheable read transaction which returns a data source of local cluster or peer CPU.
0x0060	0x-	BUS_ACCESS_RD	Bus access, read. Counts for every beat of data transferred over the read data channel between the core and the SCU. This event is a sum of LLPP_ACCESS_RD, MM_ACCESS_RD, LLRAM_ACCESS_RD, SPP_ACCESS_RD and TCMS_ACCESS_RD.
0x0061	0x-	BUS_ACCESS_WR	Bus access, write. Counts for every beat of data transferred over the write data channel between the core and the SCU. This event is a sum of LLPP_ACCESS_WR, MM_ACCESS_WR, LLRAM_ACCESS_WR, SPP_ACCESS_WR and TCMS_ACCESS_WR.
0x0066	0x-	MEM_ACCESS_RD	Data memory access, read. Counts memory accesses due to load instructions. Does not count instruction fetches, cache maintenance instructions, translation table walks or prefetches.
0x0067	0x-	MEM_ACCESS_WR	Data memory access, write. Counts memory accesses due to store instructions. Does not count instruction fetches, cache maintenance instructions, translation table walks or prefetches.
0x0070	0x-	LD_SPEC	Operation speculatively executed, load.
0x0071	0x-	ST_SPEC	Operation speculatively executed, store.
0x0072	0x-	LDST_SPEC	Operation speculatively executed, load or store. This event counts the sum of LD_SPEC and ST_SPEC.
0x0073	0x-	DP_SPEC	Operation speculatively executed, integer data-processing.
0x0074	0x-	ASE_SPEC	Operation speculatively executed, Advanced SIMD instruction.
0x0075	0x-	VFP_SPEC	Operation speculatively executed, floating-point instruction.
0x0076	0x-	PC_WRITE_SPEC	Operation speculatively executed, software change of the Program Counter.
0x0078	0x-	BR_IMMED_SPEC	Branch speculatively executed, immediate branch.
0x0079	0x-	BR_RETURN_SPEC	Branch speculatively executed, procedure return.

Event code	ETM code	Mnemonic	Description
0x007A	0x-	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch.
0x0082	0x33	EXC_SVC	Exception taken, supervisor call
0x0086	0x32	EXC_IRQ	Exception taken, IRQ
0x0087	0x31	EXC_FIQ	Exception taken, FIQ
0x008A	0x30	EXC_HVC	Exception taken, Hypervisor Call
0x008E	0x2F	EXC_TRAP_IRQ	Exception taken, IRQ not taken locally
0x008F	0x2E	EXC_TRAP_FIQ	Exception taken, FIQ not taken locally

14.4.2 Cluster PMU events

The cluster *Performance Monitoring Unit* (PMU) collects events from the shared units and use event codes to reference these events. Cluster PMU events are implementation defined.

14.4.2.1 Implementation defined cluster PMU events

The following table shows the implementation defined events that are generated at the cluster and the numbers that the *Performance Monitoring Unit* (PMU) uses to reference the events.

Table 14-6: Implementation defined cluster PMU events

Event code	ETM code	Mnemonic	Description
0x0419	0x-	MM_ACCESS	Main Manager bus access counter. This event is a sum of MM_ACCESS_RD and MM_ACCESS_WR.
0x041A	0x-	MM_ACTIVE	Main Manager bus has outstanding transactions (counted on SCLK instead of bus clock)
0x041D	0x-	MM_CYCLES	Main Manager bus cycles
0x0460	0x-	MM_ACCESS_RD	Main Manager access, read. Counts for every beat of data transferred over the read data channel between the cluster and the interconnect.
0x0461	0x-	MM_ACCESS_WR	Main Manager access, write. Counts for every beat of data transferred over the write data channel between the cluster and the interconnect.
0x0462	0x-	MM_ACCESS_SHARED	Main Manager access, shared. Counts for every beat of shared data transferred over the data channels between the cluster and the interconnect.
0x0463	0x-	MM_ACCESS_NOT_SHARED	Main Manager access, not shared. Counts for every beat of not shared data transferred over the data channels between the cluster and the interconnect.
0x0464	0x-	MM_ACCESS_NORMAL	Main Manager access, normal. Counts for every beat of normal data transferred over the data channels between the cluster and the interconnect.
0x0465	0x-	MM_ACCESS_PERIPH	Main Manager access, periph. Counts for every beat of device data transferred over the data channels between the cluster and the interconnect.
0x0466	0x-	MM_CHI_SNP_ACCESS	L2 external snoop access counter. Counts every external snoop request.
0x0467	0x-	MM_CHI_SNP_EVICT	L2 external snoop eviction counter. Counts every external snoop request that causes an L2 cache eviction.
0x0468	0x-	MM_CHI_SNP_NO_CPU	L2 external No-cpu snoop access counter. Counts every external snoop request that completes without snooping a core.

Event code	ETM code	Mnemonic	Description
0x0469	0x-	MM_PREFETCH_CPU_ACCESS	L2 prefetch access, CPU counter. Counts every stash transaction originating from a core.
0x0470	0x-	MM_PREFETCH_CPU_HIT	L2 prefetch data hit, CPU counter. Counts every stash transaction originating from a core where the stash hit in the cluster.
0x0471	0x-	MM_PREFETCH_CPU_MISS	L2 prefetch data miss, CPU counter. Counts every stash transaction originating from a core where data was read in from outside the cluster.
0x0472	0x-	MM_PREFETCH_CPU_MATCH	L2 prefetch matching access, CPU counter. Counts every completed stash transaction originating from a core that is matched by a compatible read request.
0x0473	0x-	MM_PREFETCH_CPU_KILL	L2 prefetch terminated access, CPU counter. Counts every stash transaction originating from a core that is terminated due to an incompatible match.
0x0474	0x-	MM_CHI_STASH_ICN_ACCESS	L2 stash access, ICN counter. Counts every stash transaction originating from the interconnect.
0x0475	0x-	MM_CHI_STASH_ICN_HIT	L2 stash data hit, ICN counter. Counts every stash transaction originating from the interconnect where the stash hit in the cluster.
0x0476	0x-	MM_CHI_STASH_ICN_MISS	L2 stash data miss, ICN counter. Counts every stash transaction originating from the interconnect where data was read in from outside the cluster.
0x0477	0x-	MM_CHI_STASH_ICN_MATCH	L2 stash matching access, ICN counter. Counts every completed stash transaction originating from the interconnect that is matched by a compatible read request.
0x0478	0x-	MM_CHI_STASH_ICN_KILL	L2 stash terminated access, ICN counter. Counts every stash transaction originating from the interconnect that is terminated due to an incompatible match.
0x0484	0x-	MM_HAZARD_ADDR	L2 address hazard. Request flushed and replayed due to address match with earlier request.
0x0485	0x-	MM_HAZARD_L2DB	L2 data buffer hazard. Request flushed and replayed due to L2DBs not available.
0x0486	0x-	MM_HAZARD_AFB	L2 address forwarding buffer hazard. Request flushed and replayed due to AFB not available.
0x0487	0x-	MM_HAZARD_STU_DRAIN	L2 STU drain hazard. Read after Write hazard or Data Buffers full requiring force drain of the STU.
0x0488	0x-	MM_MEMORY_ERROR	Local memory error. Counts any Correctable or Uncorrectable memory error (ECC or parity) in the Duplicate Tags, L2 Tag or Data and L2DBs.
0x0119	0x-	MACP_ACCESS	Main Accelerator Coherency Port (MACP) bus access counter. This event is a sum of MACP_ACCESS_RD and MACP_ACCESS_WR.
0x011A	0x-	MACP_ACTIVE	MACP bus has outstanding transactions (counted on SCLK instead of bus clock)
0x011D	0x-	MACP_CYCLES	MACP bus cycles
0x0160	0x-	MACP_ACCESS_RD	MACP bus read access counter. Counts for every beat transferred over the read data channel between the interconnect and the cluster.
0x0161	0x-	MACP_ACCESS_WR	MACP bus write access counter. Counts for every beat transferred over the write data channel between the interconnect and the cluster.
0x0162	0x-	MACP_STASH_ACP_ACCESS	L2 stash access, ACP counter. Counts every stash transaction originating from the MACP.
0x0163	0x-	MACP_STASH_ACP_HIT	L2 stash data hit, ACP counter. Counts every stash transaction originating from the MACP where the stash hit in the cluster.
0x0164	0x-	MACP_STASH_ACP_MISS	L2 stash data miss, ACP counter. Counts every stash transaction originating from the MACP where data was read in from outside the cluster.
0x0165	0x-	MACP_STASH_ACP_MATCH	L2 stash matching access, ACP counter. Counts every completed stash transaction originating from the MACP that is matched by a compatible read request.

Event code	ETM code	Mnemonic	Description
0x0166	0x-	MACP_STASH_ACP_KILL	L2 stash terminated access, ACP counter. Counts every stash transaction originating from the MACP that is terminated due to an incompatible match.
0x0319	0x-	ACELS_ACCESS	ACE-Lite Subordinate (ACELS) bus access counter. This event is a sum of ACELS_ACCESS_RD and ACELS_ACCESS_WR.
0x031D	0x-	ACELS_CYCLES	ACELS bus cycles
0x0360	0x-	ACELS_ACCESS_RD	ACELS bus read access counter. This event counts accesses on the read address channel.
0x0361	0x-	ACELS_ACCESS_WR	ACELS bus write access counter. This event counts accesses on the write address channel.
0x0362	0x-	ACELS_HAZARD_ID_RD	ACELS request stalled due to ID hazard on the read address channel
0x0363	0x-	ACELS_HAZARD_ID_WR	ACELS request stalled due to ID hazard on the write address channel.
0x0364	0x-	ACELS_HAZARD_RESP_RD	ACELS response on the read channel delayed due to arbitration contention.
0x0365	0x-	ACELS_HAZARD_RESP_WR	ACELS response on the write channel delayed due to arbitration contention.
0x0219	0x-	SPP_ACCESS	Shared Peripheral Port bus access counter. This event is a sum of SPP_ACCESS_RD and SPP_ACCESS_WR.
0x021A	0x-	SPP_ACTIVE	Shared Peripheral Port bus has outstanding transactions (counted on SCLK instead of bus clock).
0x021D	0x-	SPP_CYCLES	Shared Peripheral Port cycles
0x0260	0x-	SPP_ACCESS_RD	Shared Peripheral Port access, read. Counts for every beat of data transferred over the read data channel on the port.
0x0261	0x-	SPP_ACCESS_WR	Shared Peripheral Port access, write. Counts for every beat of data transferred over the write data channel on the port.
0x0519	0x-	L2_CACHE	L2 unified cache access counter. Counts every cacheable read or write transaction issued to the L2. This event is a sum of L2_CACHE_RD and L2_CACHE_WR.
0x0520	0x-	L2_CACHE_RD	L2 unified cache access, read counter. Counts every cacheable read transaction issued to the L2 (excluding prefetches and stashes).
0x0521	0x-	L2_CACHE_WR	L2 unified cache access, write counter. Counts every cacheable write transaction issued to the L2.
0x0522	0x-	L2_CACHE_REFILL	L2 unified cache access refill counter. Counts every cacheable read or write transaction issued to the interconnect. This event is a sum of L2_CACHE_REFILL_RD and L2_CACHE_REFILL_WR.
0x0523	0x-	L2_CACHE_REFILL_RD	L2 unified cache access, read refill counter. Counts every cacheable read transaction issued to the interconnect (excluding prefetches and stashes).
0x0524	0x-	L2_CACHE_REFILL_WR	L2 unified cache access, write refill counter. Counts every cacheable write transaction issued to the interconnect.
0x0525	0x-	L2_CACHE_WB	L2 unified cache writeback counter. Counts every write-back from the L2 cache.
0x0526	0x-	L2_CACHE_ALLOCATE	L2 unified cache allocation without refill counter. Counts every full cache line write into the L2 cache which does not cause a linefill.
0x0719	0x-	LLRAM_ACCESS	Low-latency RAM (LLRAM) bus access counter. This event is a sum of LLRAM_ACCESS_RD and LLRAM_ACCESS_WR.
0x071A	0x-	LLRAM_ACTIVE	LLRAM bus has outstanding transactions (counted on SCLK instead of bus clock).
0x071D	0x-	LLRAM_CYCLES	LLRAM bus cycles
0x0760	0x-	LLRAM_ACCESS_RD	Bus access, read. Counts for every beat of data transferred over the read data channel between the cluster and the interconnect.

Event code	ETM code	Mnemonic	Description
0x0761	0x-	LLRAM_ACCESS_WR	Bus access, write. Counts for every beat of data transferred over the write data channel between the cluster and the interconnect.
0x0762	0x-	LLRAM_ACCESS_SHARED	Bus access, shared. Counts for every beat of shared data transferred over the data channels between the cluster and the interconnect.
0x0763	0x-	LLRAM_ACCESS_NOT_SHARED	Bus access, not shared. Counts for every beat of not shared data transferred over the data channels between the cluster and the interconnect.
0x0764	0x-	LLRAM_ACCESS_NORMAL	Bus access, normal. Counts for every beat of normal data transferred over the data channels between the cluster and the interconnect.
0x0765	0x-	LLRAM_ACCESS_PERIPH	Bus access, periph. Counts for every beat of device data transferred over the data channels between the cluster and the interconnect.
0x0766	0x-	LLRAM_HAZARD_ADDR	LCU address hazard. Request ordered due to address match with earlier request.
0x0767	0x-	LLRAM_HAZARD_SET_WAY	LCU set/way hazard. Request ordered due to set/way match with earlier request.
0x0768	0x-	LLRAM_HAZARD_STU_DRAIN	LCU STU drain hazard. Read after Write hazard or LCU Data Buffers full requiring LCU force drain of the STU.
0x0769	0x-	LLRAM_MEMORY_ERROR	Local memory error. Counts any Correctable or Uncorrectable memory error (ECC or parity) in the Duplicate Tags.
0x0819	0x-	ROUTER_STALL	More than one router packet needs arbitrating introducing contention.

14.4.2.2 Recommended implementation defined cluster PMU events

Arm recommends some implementation defined PMU events at the cluster.

The following table shows the recommended implementation defined events that are generated at the cluster and the numbers that the *Performance Monitoring Unit* (PMU) uses to reference the events.

Table 14-7: Recommended implementation defined cluster PMU events

Event code	ETM code	Mnemonic	Description
0x0011	0x-	CYCLES	Clock cycles
0x0019	0x-	BUS_ACCESS	Bus access, read or write. Counts for every beat of data transferred over the read or write data channel between the cluster and the interconnect.
0x001A	0x-	MEMORY_ERROR	Local memory error. Counts any Correctable or Uncorrectable memory error (ECC or parity) in the protected RAMs.
0x001D	0x-	BUS_CYCLES	Bus clock cycle
0x001E	0x-	CHAIN	Odd performance counter chain mode
0x0060	0x-	BUS_ACCESS_RD	Bus access, read. Counts for every beat of data transferred over the read data channel between the cluster and the interconnect.
0x0061	0x-	BUS_ACCESS_WR	Bus access, write. Counts for every beat of data transferred over the write data channel between the cluster and the interconnect.
0x0062	0x-	BUS_ACCESS_SHARED	Bus access, shared. Counts for every beat of shared data transferred over the data channels between the cluster and the interconnect.
0x0063	0x-	BUS_ACCESS_NOT_SHARED	Bus access, not shared. Counts for every beat of not shared data transferred over the data channels between the cluster and the interconnect.

Event code	ETM code	Mnemonic	Description
0x0064	0x-	BUS_ACCESS_NORMAL	Bus access, normal. Counts for every beat of normal data transferred over the data channels between the cluster and the interconnect.
0x0065	0x-	BUS_ACCESS_PERIPH	Bus access, periph. Counts for every beat of device data transferred over the data channels between the cluster and the interconnect.

14.5 PMU interrupts

The Cortex®-R82 processor asserts nCLUSTERPMUIRQ and nPMUIRQ signals when the PMU generates an interrupt.

The Cortex®-R82 processor asserts:

- The nCLUSTERPMUIRQ signal when the cluster PMU generates an interrupt.
- The nPMUIRQ[k] signal where k is the core ID, when the per-core PMU generates an interrupt.

You can route nCLUSTERPMUIRQ and nPMUIRQ signals to an external interrupt controller for prioritization and masking. This is the only mechanism that signals these interrupts to a core. nCLUSTERPMUIRQ and nPMUIRQ interrupts are also driven as a trigger input to the cluster *Cross Trigger Interface* (CTI) and core CTI respectively.

14.6 PMU register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary table for the Cortex®-R82 processor AArch64 Performance Monitors registers in [A.1.3 AArch64 Performance Monitors registers summary](#) on page 285.

You can find the register summary table for the core external Performance Monitors registers in [B.2.1.8 External PMU registers summary](#) on page 1613 and cluster external Performance Monitors registers in [B.2.1.2 External CLUSTERPMU registers summary](#) on page 1602.

15. ETM

This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex®-R82 processor.

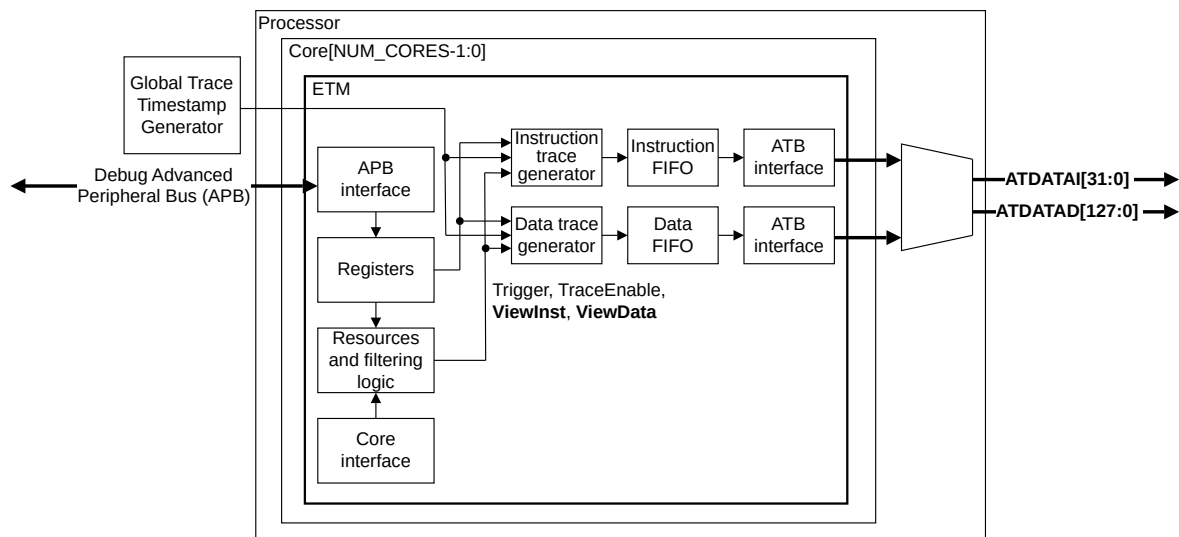
15.1 About the ETM

The *Embedded Trace Macrocell* (ETM) performs real-time instruction and data flow tracing based on the ETM architecture ETMv4.5. The Cortex®-R82 processor supports one ETM per core.

The ETM is a CoreSight™ component and is an integral part of the Arm Real-time Debug solution, DS-5 Development Studio. It enables non-invasive debugging of software running on one or more cores of the Cortex®-R82 processor.

The following figure shows the main functional blocks of the ETM.

Figure 15-1: ETM block diagram



Core interface

This block connects to the core within the Cortex®-R82 processor. It tracks the execution information from the core, decodes the control signals, and passes on the information to the internal interfaces.

Instruction trace generator

This block generates the trace packets which are a compressed form of the instruction execution information provided by the core. The trace packets are then passed to the instruction FIFO.

Data trace generator

This block generates the trace packets which are a compressed form of the data transfers (data address and data value) provided by the core. The trace packets are then passed to the data FIFO.

FIFO

This block buffers bursts of trace packets. Separate FIFOs are provided, one for the instruction trace stream, and one for the data trace stream.

Resources and filtering logic

These blocks contain resources which the trace software programs to trigger and filter the trace information. They start and stop trace generation, depending on the conditions that have been set.

ATB interfaces

There are two ATB interfaces:

- *Instruction ATB interface:* This reads up to four bytes of compressed packet information from the instruction FIFO and sends them over the instruction ATB interface. All the core ETM instruction trace streams are funneled into a single 32-bit ATB trace bus.
- *Data ATB interface:* This reads up to eight bytes of compressed packet information from the data FIFO and sends them over the data ATB interface. All the core ETM data trace streams and all the ELA trace streams are funneled into a single 128-bit ATB trace bus.

APB interface

This block implements the interface to the APB that provides access to the programmable registers.

Global timestamping

The ETM reuses the generic timer CNTVALUEB as a reference for timestamp packets.

This provides a 64-bit timestamp that a debugger can use for coarse-grained profiling, and correlation of trade sources.



Decompression of data trace relies on the presence of a global timestamp count.

15.2 ETM trace unit generation options and resources

The following table shows the resources of the ETM that are implemented.

Table 15-1: ETM trace unit resources implemented

Feature	Instance
Address comparators	4 pairs
Data value comparators	2

Feature	Instance
Context ID comparators	1
Virtual machine ID comparators	1
Single-Shot comparator resource	2
Counters	2
Cycle count size	12 bits
Number of sequencer states	4
Processor comparator inputs	0
External inputs	58
External outputs	4
External input selectors	4
Resource selector pairs	8
Instruction trace port size	32-bit
Data trace port size	64-bit
Instruction FIFO ⁶	128 bytes with 32-bit output
Data FIFO	256 bytes with 64-bit output
Claim tag bits	4

The following table shows the optional features of the ETM architecture that the ETM implements.

Table 15-2: ETM trace unit generation options implemented

Feature	Implemented
Trace Start/Stop block	Yes
Trace all branches option	Yes
Trace of conditional instructions	Yes
Cycle counting in instruction trace	Yes
Data trace supported	Yes
Data address comparison	Yes
OS Lock mechanism	Yes
Secure non-invasive debug	Yes
Context ID tracing	Yes
Trace output	Yes
Timestamp size	64-bit
Memory mapped access to ETM registers	No
External debugger access to ETM registers	Yes
System instruction access to ETM registers	Yes
VMID comparator support	Yes
ATB trigger support	Yes

⁶ Instruction trace can be configured to take priority over data trace. See bit[10] of the TRCSTALLCTLR.

15.3 ETM event connectivity

This section describes how the Cortex®-R82 processor *Embedded Trace Macrocell* (ETM) inputs and outputs are connected to the *Cross Trigger Interface* (CTI) and *Performance Monitoring Unit* (PMU).

The following table shows the connection of the ETM external inputs that come from the CTI and PMU.

Table 15-3: ETM External Input connections

Bits	Description
ETM External Input 0	CTI Trigger Output 4
ETM External Input 1	CTI Trigger Output 5
ETM External Input 2	CTI Trigger Output 6
ETM External Input 3	CTI Trigger Output 7

The ETM external output resources are connected to the CTI, as the following table shows.

Table 15-4: ETM External Output connections to CTI

ETM output	CTI input
ETM External Output 0	CTI Trigger Input 4
ETM External Output 1	CTI Trigger Input 5
ETM External Output 2	CTI Trigger Input 6
ETM External Output 3	CTI Trigger Input 7

15.4 Operation

This section describes the R82 processor *Embedded Trace Macrocell* (ETM) **IMPLEMENTATION DEFINED** features.

See the [Arm® Embedded Trace Macrocell Architecture Specification ETMv4.0 to ETM4.6](#) for more information on the operation.

15.4.1 Precise TraceEnable events

The ViewInst and ViewData are imprecise under certain conditions, with some implementation-defined exceptions. The only condition which ensures that ViewInst and ViewData are precise is that the enabling event condition is TRUE.

For more information, see the [Arm® Embedded Trace Macrocell Architecture Specification ETMv4.0 to ETM4.6](#).

15.4.2 Parallel instruction execution

The Cortex®-R82 processor supports parallel instruction execution. The macrocell can trace up to three instructions per cycle and up to 256 bits of data transfer per cycle.

If ViewInst is active for a cycle, the ETM:

- Always traces all instructions reported to the ETM on the same cycle up to a branch if present.
- Any instructions from the previous cycle that follow a branch instruction.
- Can trace data for any of the instructions traced.

15.4.3 Comparator features

The ETM implements data address comparison. There are eight address comparators that can be configured for either instruction or data address comparison.

The ViewData instruction address comparators are sensitive to a batch of instructions which execute in consecutive cycles. If any instruction in the batch is in an include range and any instruction is not in an exclude range, then ViewData can be high.

See the [Arm® Embedded Trace Macrocell Architecture Specification ETMv4.0 to ETM4.6](#) for a description of data address comparison.

15.4.4 Trace features

The ETM implements all of the ETMv4.5 trace features.

This means it supports:

- Data value and data address tracing.
- Data suppression.
- Cycle-accurate tracing.
- Timestamping.

See the [Arm® Embedded Trace Macrocell Architecture Specification ETMv4.0 to ETM4.6](#) for a description of these features.

15.4.5 Packet formats

The Cortex®-R82 processor ETM instruction trace interface does not support the following packet formats:

- Speculation resolution:
 - Mispredict packet.

- Cancel format 2 and 3 packets.
- Conditional tracing:
 - All instruction format packets.
 - Result format 1 packet.
- Q packets.

The Cortex®-R82 processor ETM data trace interface supports all data trace packet types except the P1 Format 6 and 7 packets.

See the [Arm® Embedded Trace Macrocell Architecture Specification ETMv4.0 to ETM4.6](#) for the trace packet format descriptions.

15.4.6 Resource selection

The ETM uses event selectors to control resources.

The ETM controls the following resources:

- Trace events (triggers and markers in the trace stream).
- Timestamp event.
- ViewInst event.
- ViewData event.
- Counter control.
- Sequencer state transitions.

Each event selector is configured to be sensitive to a resource selector pair, and one resource selector pair can control more than one event selector.

The ETM provides one fixed resource selector pair, with static values of 0 and 1, and seven configurable selector pairs. A resource selector pair provides a bitfield OR selector for resources in two different groups, with each group and a configurable boolean combination provided.

The following shows the resources that can be selected for the instruction and data trace.

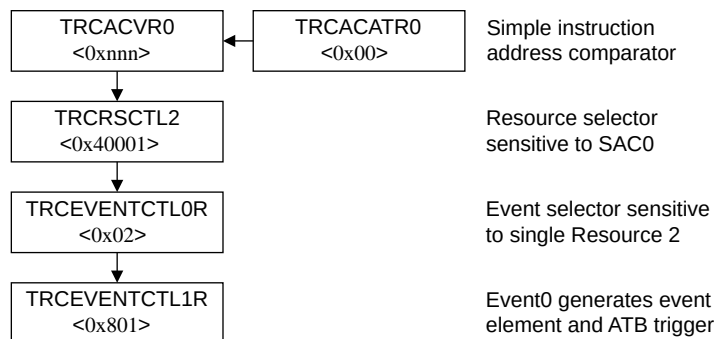
Table 15-5: Instruction and data resource selection

Group	Select	Resource
0b0000	0-3	External input selector 0-3
0b0010	0-1	Counter at zero 0-1
	4-7	Sequencer states 0-3
0b0011	1	Single-Shot comparator 0-1
0b0100	0-7	Single address comparator 0-7
0b0101	0-3	Address range comparator 0-3

Group	Select	Resource
0b0110	0	Context ID Comparator 0
0b0111	0	VMID Comparator 0

For example, the following figure shows the steps necessary to use a single address comparator to generate a trigger event and an ATB trigger. This example uses the first single resource selector that can be user-configured.

Figure 15-2: Trigger event resource selection



15.4.7 Trace flush behavior

Events that are observed by the ETM can be confirmed to have reached the trace bus output with the use of the ATB flush protocol. Both ATB ports must be flushed to determine when the trace infrastructure finishes capturing all the packets generated by the ETM.

ETM internally flushes instruction and data trace together whenever either flush request is seen but does not guarantee that the trace data has drained from the ETM. When the processor enters a low-power state, all trace data is output from the ETM.

15.4.8 Low-power state behavior

When the Cortex®-R82 processor enters a low-power state, there is a delay before the resources in the ETM become inactive.

This permits the last instruction executed to trigger a comparator, update the counter or sequencer, and the resultant event packet to be inserted in the specified trace stream. This event packet is presented on the trace bus before the ETM itself enters a low-power state.

If an event packet is generated for a different reason, it is not guaranteed to be output before the ETM enters a low-power state, but is traced when the processor leaves the low-power state, if the ETM logic is not reset before this can occur.

The TRCEVENTCTL1R.LPOVERRIDE bit controls how a trace unit behaves in a low-power state. If it is set to 1, trace unit low-power state behavior is overridden, that is, entry to a low-power state does not affect the trace unit resources or trace generation. In this case, the ETM resources remain active.

15.4.9 Cycle counter

The Cortex®-R82 processor ETM uses a 12-bit cycle counter.

The ETM cycle counter does not count when the Cortex®-R82 processor is in a low-power state.

15.4.10 Non-architectural exceptions

Non-architectural behavior exceptions are indicated by the ETM.

The ETM indicates exceptions for the following non-architectural behavior that use the following TYPE encoding:

0b10001	ECC logic requires instruction trace to be replayed. This should not be relied on as providing trace of ECC behavior.
----------------	---

15.4.11 Trace synchronization

The ETM receives and combines all sources of trace synchronization requests to determine when synchronization is required. When synchronization is required, information is inserted in both trace streams as necessary (depending on whether data trace is active).

To decompress the trace streams, synchronization information in the data trace stream determines the alignment with synchronization information in the instruction stream. If the ETM is configured to trace only events in the data stream, you must configure the instruction trace stream to contain sufficient elements to permit the required data trace stream synchronization.

15.5 Modes of operation and execution

This section describes how to control the ETM programming and read and program the ETM registers.

15.5.1 Use of the ETM main enable bit

When programming the ETM registers, you must enable all the changes at the same time.

For example, if the counter is reprogrammed, it might start to count based on incorrect events, before the trigger condition has been correctly set up.

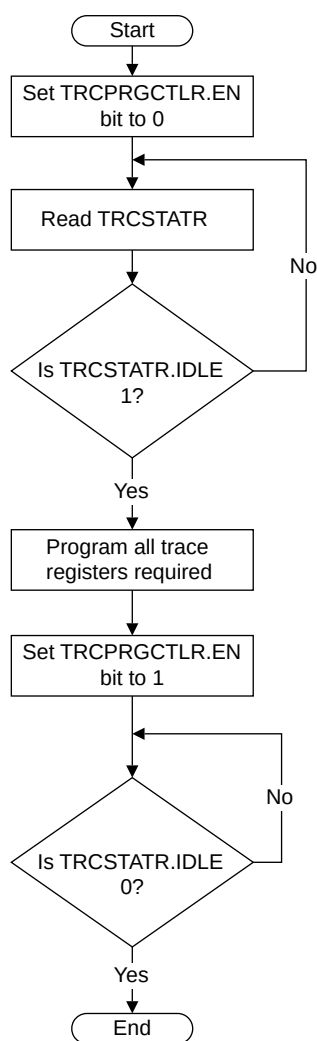
To disable all trace operations during programming use:

- The ETM main enable in the TRCPRGCTLR.
- The TRCSTATR to indicate the ETM status.

The Cortex®-R82 processor does not have to be in Debug state while you program the ETM registers.

The following figure shows the procedure to use.

Figure 15-3: Programming ETM registers



15.5.2 Programming and reading ETM registers

To access the ETM registers, use the system register access. This provides a direct method of programming the ETM.

15.5.3 External register access permissions

Whether access is permitted to a register depends on:

- If the processor is powered up.
- The state of the OS Lock.
- The state of the debug authentication inputs to the processor.

The behavior that is specific to each register and the type of access to the register is not described in this document. For a detailed description of these features and their effects on the registers, see the [Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture](#).

The register descriptions provided in this section describe whether each register is read/write or read-only.

15.6 ETM register summary

The register summary tables provide an overview of all the relevant registers.

You can find the register summary table for the Cortex®-R82 processor AArch64 *Embedded Trace Macrocell* (ETM) registers in [A.1.8 AArch64 Trace unit registers summary](#) on page 298.

You can find the register summary table for the core external ETM registers in [B.2.1.7 External ETM registers summary](#) on page 1609.

16. Advanced SIMD and floating-point support

This chapter introduces the optional Advanced SIMD and floating-point support.

16.1 About the Advanced SIMD and floating-point support

The Cortex®-R82 processor supports the Advanced SIMD and scalar floating-point instructions in the Arm®v8-R AArch64 instruction set without floating-point exception trapping. The Cortex®-R82 processor floating-point implementation includes several features from Armv8.2 and Arm®v8.3.

For more information on the architectural features that are implemented by the Cortex®-R82 processor see, [3.2.1 Architecture requirements](#) on page 39.

Each core within the Cortex®-R82 processor has optional Advanced SIMD and floating-point support.

The Cortex®-R82 processor implements all scalar and vector operations in hardware with support for all combinations of:

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

The Arm®v8-R AArch64 architecture does not define a separate version number for its Advanced SIMD and floating-point support because the instructions are always implicitly present.

16.2 Low-latency single-precision instructions

The Cortex®-R82 processor has a per-core `LOW_LATENCY_SP` parameter that controls the inclusion of additional hardware that exclusively processes scalar single-precision instructions.

If the `LOW_LATENCY_SP` is set to 0, the Advanced SIMD and scalar floating-point instructions complete in five cycles. If the `LOW_LATENCY_SP` is set to 1, an additional hardware is included that enables the scalar single-precision instructions to complete in three cycles.



Division and square root instructions take longer to complete. Scalar single-precision divisions and square roots are not sped up when `LOW_LATENCY_SP` is set to 1.

Including this feature enables your program to execute faster if and when using single-precision instructions.

The following table shows the scalar single-precision instructions that execute in three cycles when the `LOW_LATENCY_SP = 1`.

Table 16-1: Low-latency scalar single-precision instructions

Instruction	Conditions
FABD	sz == 0b0
FADD	ftype == 0b00
FMADD	ftype == 0b00
FMSUB	ftype == 0b00
FMUL	ftype == 0b00
FMULX	sz == 0b0
FNMADD	ftype == 0b00
FNMSUB	ftype == 0b00
FNMUL	ftype == 0b00
FRECPE	sz == 0b0
FRECPS	sz == 0b0
FRECPX	sz == 0b0
FRINTA/FRINTM/FRINTP/FRINTI/FRINTX/FRINTZ	ftype == 0b00
FRSQRT	sz == 0b0
FRSQRTS	sz == 0b0
FSUB	ftype == 0b00
SCVTF/UCVTF	ftype == 0b00

Appendix A AArch64 registers

This appendix contains the descriptions for all the AArch64 registers in the Cortex®-R82 processor.

A.1 AArch64 register summaries

This section includes the register summary tables for all the AArch64 registers in the Cortex®-R82 processor.

A.1.1 AArch64 Identification registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Identification 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-1: 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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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
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_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
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
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.1.2 AArch64 Generic System Control registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-2: Generic System Control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
MPUIR_EL1	3	0	C0	C0	4	See individual bit resets.	64-bit	MPU Type Register (EL1)
SCTLR_EL1	3	0	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL1)
ACTLR_EL1	3	0	C1	C0	1	See individual bit resets.	64-bit	Auxiliary Control Register (EL1)
CPACR_EL1	3	0	C1	C0	2	See individual bit resets.	64-bit	Architectural Feature Access 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])
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)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
ESR_EL1	3	0	C5	C2	0	See individual bit resets.	64-bit	Exception Syndrome Register (EL1)
PRBAR<n>_EL1	3	0	C6	1:m[3:1]	m[0]:00	See individual bit resets.	64-bit	Protection Region Base Address Register n (EL1)
PRLAR<n>_EL1	3	0	C6	1:m[3:1]	m[0]:01	See individual bit resets.	64-bit	Protection Region Limit Address Register n (EL1)
FAR_EL1	3	0	C6	C0	0	See individual bit resets.	64-bit	Fault Address Register (EL1)
PRENR_EL1	3	0	C6	C1	1	See individual bit resets.	64-bit	Protection Region Enable Register (EL1)
PRSELR_EL1	3	0	C6	C2	1	See individual bit resets.	64-bit	Protection Region Selection Register (EL1)
PRBAR_EL1	3	0	C6	C8	0	See individual bit resets.	64-bit	Protection Region Base Address Register (EL1)
PRLAR_EL1	3	0	C6	C8	1	See individual bit resets.	64-bit	Protection Region Limit 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)
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
IMP_ITCMREGIONR_EL1	3	0	C15	C0	1	See individual bit resets.	64-bit	ITCM Region Register
IMP_DTCMREGIONR_EL1	3	0	C15	C0	2	See individual bit resets.	64-bit	DTCM Region Register
IMP_LLPPREGIONR_EL1	3	0	C15	C0	3	See individual bit resets.	64-bit	LLPP Region Register
IMP_LLRAMREGIONR_EL1	3	0	C15	C0	4	See individual bit resets.	64-bit	LLRAM Region Register
IMP_SPPREGIONR_EL1	3	0	C15	C0	5	See individual bit resets.	64-bit	SPP Region Register
IMP_CPUACTLR_EL1	3	0	C15	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register
IMP_BPCTLR_EL1	3	0	C15	C1	1	See individual bit resets.	64-bit	Branch Predictor Control Register
IMP_MEMPROTCTLR_EL1	3	0	C15	C1	5	See individual bit resets.	64-bit	Memory Protection Control Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPUCFR_EL1	3	0	C15	C2	0	See individual bit resets.	64-bit	CPU Configuration Register
IMP_CPUPWRCTLR_EL1	3	0	C15	C2	7	See individual bit resets.	64-bit	CPU Power Control Register
IMP_CLUSTERCFR_EL1	3	0	C15	C3	0	See individual bit resets.	64-bit	Cluster Configuration Register
IMP_CLUSTERACTLR_EL1	3	0	C15	C3	3	See individual bit resets.	64-bit	Cluster Auxiliary Control Register
IMP_CLUSTERPWRCTLR_EL1	3	0	C15	C3	5	See individual bit resets.	64-bit	Cluster Power Control Register
IMP_CLUSTERPWRDN_EL1	3	0	C15	C3	6	See individual bit resets.	64-bit	Cluster Power Down Register
IMP_CLUSTERPWRSTAT_EL1	3	0	C15	C3	7	See individual bit resets.	64-bit	Cluster Power Status Register
IMP_CLUSTERSID_EL1	3	0	C15	C4	0	See individual bit resets.	64-bit	Cluster Scheme ID Register
IMP_CLUSTERACPSID_EL1	3	0	C15	C4	1	See individual bit resets.	64-bit	Cluster ACP Scheme ID Register
IMP_CLUSTERPARTCR_EL1	3	0	C15	C4	3	See individual bit resets.	64-bit	Cluster Partition Control Register
IMP_CLUSTERQOSR_EL1	3	0	C15	C4	4	See individual bit resets.	64-bit	Cluster Quality of Service Register
IMP_CLUSTERACELSCTLR_EL1	3	0	C15	C4	5	See individual bit resets.	64-bit	ACELS Port Control Register
IMP_CLUSTERMEMPROTCTLR_EL1	3	0	C15	C4	6	See individual bit resets.	64-bit	Cluster Memory Protection Control Register
AIDR_EL1	3	1	C0	C0	7	See individual bit resets.	64-bit	Auxiliary ID Register
IMP_CDBGDR0_EL1	3	2	C15	C0	0	See individual bit resets.	64-bit	Cache Debug Data Register 0
IMP_CDBGDR1_EL1	3	2	C15	C0	1	See individual bit resets.	64-bit	Cache Debug Data Register 1
IMP_CLUSTERCDBGDR0_EL1	3	2	C15	C3	0	See individual bit resets.	64-bit	Cluster Cache Debug Data Register 0
IMP_CPUPSELR_EL1	3	2	C15	C8	0	See individual bit resets.	64-bit	Instruction Patch Selection Register
IMP_CPUPCR_EL1	3	2	C15	C8	1	See individual bit resets.	64-bit	Selected Instruction Patch Control Register
IMP_CPUPCR<n>_EL1	3	2	C15	C8	1	See individual bit resets.	64-bit	Instruction Patch Control Registers
IMP_CPUPOR_EL1	3	2	C15	C8	2	See individual bit resets.	64-bit	Selected Instruction Patch Opcode Register
IMP_CPUPOR<n>_EL1	3	2	C15	C8	2	See individual bit resets.	64-bit	Instruction Patch Opcode Registers
IMP_CPUPMR_EL1	3	2	C15	C8	3	See individual bit resets.	64-bit	Selected Instruction Patch Mask Register

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IMP_CPUPMR<n>_EL1	3	2	C15	C8	3	See individual bit resets.	64-bit	Instruction Patch Mask Registers
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
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_ELO	3	3	C13	C0	2	See individual bit resets.	64-bit	ELO Read/Write Software Thread ID Register
TPIDRRO_ELO	3	3	C13	C0	3	See individual bit resets.	64-bit	ELO Read-Only Software Thread ID Register
MPUIR_EL2	3	4	C0	C0	4	See individual bit resets.	64-bit	MPU Type Register (EL2)
ACTLR_EL2	3	4	C1	C0	1	See individual bit resets.	64-bit	Auxiliary 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)
HACR_EL2	3	4	C1	C1	7	See individual bit resets.	64-bit	Hypervisor Auxiliary Control Register
VSCTLR_EL2	3	4	C2	C0	0	See individual bit resets.	64-bit	Virtualization System Control Register (EL2)
TCR_EL2	3	4	C2	C0	2	See individual bit resets.	64-bit	Translation Control Register (EL2)
VTCR_EL2	3	4	C2	C1	2	See individual bit resets.	64-bit	Virtualization Translation Control 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)
PRBAR<n>_EL2	3	4	C6	1:m[3:1]	m[0]:00	See individual bit resets.	64-bit	Protection Region Base Address Register n (EL2)
PRLAR<n>_EL2	3	4	C6	1:m[3:1]	m[0]:01	See individual bit resets.	64-bit	Protection Region Limit Address Register n (EL2)

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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
PRENR_EL2	3	4	C6	C1	1	See individual bit resets.	64-bit	Protection Region Enable Register (EL2)
PRSELR_EL2	3	4	C6	C2	1	See individual bit resets.	64-bit	Protection Region Selection Register (EL2)
PRBAR_EL2	3	4	C6	C8	0	See individual bit resets.	64-bit	Protection Region Base Address Register (EL2)
PRLAR_EL2	3	4	C6	C8	1	See individual bit resets.	64-bit	Protection Region Limit Address Register (EL2)
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)
RVBAR_EL2	3	4	C12	C0	1	See individual bit resets.	64-bit	Reset Vector Base Address Register (if EL3 not implemented)
RMR_EL2	3	4	C12	C0	2	See individual bit resets.	64-bit	Reset Management 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
IMP_INTLATENCY_EL2	3	4	C15	C1	7	See individual bit resets.	64-bit	Interrupt Latency Register

A.1.3 AArch64 Performance Monitors registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Performance Monitors 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-3: 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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMMIR_EL1	3	0	C9	C14	6	See individual bit resets.	64-bit	Performance Monitors Machine Identification Register
IMP_CLUSTERPMCR_EL1	3	0	C15	C5	0	See individual bit resets.	64-bit	Cluster Performance Monitors Control Register
IMP_CLUSTERPMCNTENSET_EL1	3	0	C15	C5	1	See individual bit resets.	64-bit	Cluster Performance Monitors Count Enable Set register
IMP_CLUSTERPMCNTENCLR_EL1	3	0	C15	C5	2	See individual bit resets.	64-bit	Cluster Performance Monitors Count Enable Clear register
IMP_CLUSTERPMOVSSSET_EL1	3	0	C15	C5	3	See individual bit resets.	64-bit	Cluster Performance Monitors Overflow Flag Status Set register
IMP_CLUSTERPMOVSCLR_EL1	3	0	C15	C5	4	See individual bit resets.	64-bit	Cluster Performance Monitors Overflow Flag Status Clear Register
IMP_CLUSTERPMSELR_EL1	3	0	C15	C5	5	See individual bit resets.	64-bit	Cluster Performance Monitors Event Counter Selection Register
IMP_CLUSTERPMINTENSET_EL1	3	0	C15	C5	6	See individual bit resets.	64-bit	Cluster Performance Monitors Interrupt Enable Set register
IMP_CLUSTERPMINTENCLR_EL1	3	0	C15	C5	7	See individual bit resets.	64-bit	Cluster Performance Monitors Interrupt Enable Clear register
IMP_CLUSTERPMCCNTR_EL1	3	0	C15	C6	0	See individual bit resets.	64-bit	Cluster Performance Monitors Cycle Count Register
IMP_CLUSTERPMXEVTYPER_EL1	3	0	C15	C6	1	See individual bit resets.	64-bit	Cluster Performance Monitors Selected Event Type Register
IMP_CLUSTERPMXVCNTR_EL1	3	0	C15	C6	2	See individual bit resets.	64-bit	Cluster Performance Monitors Selected Event Count Register
IMP_CLUSTERPMCEID0_EL1	3	0	C15	C6	4	See individual bit resets.	64-bit	Cluster Performance Monitors Common Event Identification register 0
IMP_CLUSTERPMCEID1_EL1	3	0	C15	C6	5	See individual bit resets.	64-bit	Cluster Performance Monitors Common Event Identification register 1
IMP_CLUSTERPMCLAIMSET_EL1	3	0	C15	C6	6	See individual bit resets.	64-bit	Cluster Performance Monitors Claim Set register
IMP_CLUSTERPMCLAIMCLR_EL1	3	0	C15	C6	7	See individual bit resets.	64-bit	Cluster Performance Monitors Claim Clear register
PMCR_ELO	3	3	C9	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register
PMCNTENSET_ELO	3	3	C9	C12	1	See individual bit resets.	64-bit	Performance Monitors Count Enable Set Register
PMCNTENCLR_ELO	3	3	C9	C12	2	See individual bit resets.	64-bit	Performance Monitors Count Enable Clear Register
PMOVSCLR_ELO	3	3	C9	C12	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Clear Register
PMSWINC_ELO	3	3	C9	C12	4	See individual bit resets.	64-bit	Performance Monitors Software Increment Register
PMSELR_ELO	3	3	C9	C12	5	See individual bit resets.	64-bit	Performance Monitors Event Counter Selection Register
PMCEID0_ELO	3	3	C9	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification Register 0

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
PMCEID1_ELO	3	3	C9	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification Register 1
PMCCNTR_ELO	3	3	C9	C13	0	See individual bit resets.	64-bit	Performance Monitors Cycle Count Register
PMXEVTYPER_ELO	3	3	C9	C13	1	See individual bit resets.	64-bit	Performance Monitors Selected Event Type Register
PMXVCNTR_ELO	3	3	C9	C13	2	See individual bit resets.	64-bit	Performance Monitors Selected Event Count Register
PMUSERENR_ELO	3	3	C9	C14	0	See individual bit resets.	64-bit	Performance Monitors User Enable Register
PMOVSSET_ELO	3	3	C9	C14	3	See individual bit resets.	64-bit	Performance Monitors Overflow Flag Status Set Register
PMEVCNTR<n>_ELO	3	3	C14	10:m[4:3]	m[2:0]	See individual bit resets.	64-bit	Performance Monitors Event Count Registers
PMEVTYPER<n>_ELO	3	3	C14	11:m[4:3]	m[2:0]	See individual bit resets.	64-bit	Performance Monitors Event Type Registers
PMCCFILTR_ELO	3	3	C14	C15	7	See individual bit resets.	64-bit	Performance Monitors Cycle Count Filter Register

A.1.4 AArch64 System instructions summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-4: System instructions summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
IC_IALLUIS	1	0	C7	C1	0	See individual bit resets.	64-bit	Instruction Cache Invalidate All to PoU, Inner Shareable
IC_IALLU	1	0	C7	C5	0	See individual bit resets.	64-bit	Instruction Cache Invalidate All to PoU
DC_IVAC	1	0	C7	C6	1	See individual bit resets.	64-bit	Data or unified Cache line Invalidate by VA to PoC
DC_ISW	1	0	C7	C6	2	See individual bit resets.	64-bit	Data or unified Cache line Invalidate by Set/Way
AT_S1E1R	1	0	C7	C8	0	See individual bit resets.	64-bit	Address Translate Stage 1 EL1 Read

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
AT_S1E1W	1	0	C7	C8	1	See individual bit resets.	64-bit	Address Translate Stage 1 EL1 Write
AT_S1E0R	1	0	C7	C8	2	See individual bit resets.	64-bit	Address Translate Stage 1 ELO Read
AT_S1E0W	1	0	C7	C8	3	See individual bit resets.	64-bit	Address Translate Stage 1 ELO Write
AT_S1E1RP	1	0	C7	C9	0	See individual bit resets.	64-bit	Address Translate Stage 1 EL1 Read PAN
AT_S1E1WP	1	0	C7	C9	1	See individual bit resets.	64-bit	Address Translate Stage 1 EL1 Write PAN
DC_CSW	1	0	C7	C10	2	See individual bit resets.	64-bit	Data or unified Cache line Clean by Set/Way
DC_CISW	1	0	C7	C14	2	See individual bit resets.	64-bit	Data or unified Cache line Clean and Invalidate by Set/Way
TLBI_VMALE1OS	1	0	C8	C1	0	See individual bit resets.	64-bit	TLB Invalidate by VMID, All at stage 1, EL1, Outer Shareable
TLBI_VAE1OS	1	0	C8	C1	1	See individual bit resets.	64-bit	TLB Invalidate by VA, EL1, Outer Shareable
TLBI_ASIDE1OS	1	0	C8	C1	2	See individual bit resets.	64-bit	TLB Invalidate by ASID, EL1, Outer Shareable
TLBI_VAAE1OS	1	0	C8	C1	3	See individual bit resets.	64-bit	TLB Invalidate by VA, All ASID, EL1, Outer Shareable
TLBI_VALE1OS	1	0	C8	C1	5	See individual bit resets.	64-bit	TLB Invalidate by VA, Last level, EL1, Outer Shareable
TLBI_VAALE1OS	1	0	C8	C1	7	See individual bit resets.	64-bit	TLB Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
TLBI_RVAE1IS	1	0	C8	C2	1	See individual bit resets.	64-bit	TLB Range Invalidate by VA, EL1, Inner Shareable
TLBI_RVAAE1IS	1	0	C8	C2	3	See individual bit resets.	64-bit	TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable
TLBI_RVALE1IS	1	0	C8	C2	5	See individual bit resets.	64-bit	TLB Range Invalidate by VA, Last level, EL1, Inner Shareable

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TLBI_RVAALE1IS	1	0	C8	C2	7	See individual bit resets.	64-bit	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
TLBI_VMALE1IS	1	0	C8	C3	0	See individual bit resets.	64-bit	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable
TLBI_VAE1IS	1	0	C8	C3	1	See individual bit resets.	64-bit	TLB Invalidate by VA, EL1, Inner Shareable
TLBI_ASIDE1IS	1	0	C8	C3	2	See individual bit resets.	64-bit	TLB Invalidate by ASID, EL1, Inner Shareable
TLBI_VAAE1IS	1	0	C8	C3	3	See individual bit resets.	64-bit	TLB Invalidate by VA, All ASID, EL1, Inner Shareable
TLBI_VALE1IS	1	0	C8	C3	5	See individual bit resets.	64-bit	TLB Invalidate by VA, Last level, EL1, Inner Shareable
TLBI_VAALE1IS	1	0	C8	C3	7	See individual bit resets.	64-bit	TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
TLBI_RVAE1IOS	1	0	C8	C5	1	See individual bit resets.	64-bit	TLB Range Invalidate by VA, EL1, Outer Shareable
TLBI_RVAAE1IOS	1	0	C8	C5	3	See individual bit resets.	64-bit	TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable
TLBI_RVALE1IOS	1	0	C8	C5	5	See individual bit resets.	64-bit	TLB Range Invalidate by VA, Last level, EL1, Outer Shareable
TLBI_RVAALE1IOS	1	0	C8	C5	7	See individual bit resets.	64-bit	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
TLBI_RVAE1	1	0	C8	C6	1	See individual bit resets.	64-bit	TLB Range Invalidate by VA, EL1
TLBI_RVAAE1	1	0	C8	C6	3	See individual bit resets.	64-bit	TLB Range Invalidate by VA, All ASID, EL1
TLBI_RVALE1	1	0	C8	C6	5	See individual bit resets.	64-bit	TLB Range Invalidate by VA, Last level, EL1
TLBI_RVAALE1	1	0	C8	C6	7	See individual bit resets.	64-bit	TLB Range Invalidate by VA, All ASID, Last level, EL1
TLBI_VMALE1	1	0	C8	C7	0	See individual bit resets.	64-bit	TLB Invalidate by VMID, All at stage 1, EL1

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TLBI_VAE1	1	0	C8	C7	1	See individual bit resets.	64-bit	TLB Invalidate by VA, EL1
TLBI_ASIDE1	1	0	C8	C7	2	See individual bit resets.	64-bit	TLB Invalidate by ASID, EL1
TLBI_VAAE1	1	0	C8	C7	3	See individual bit resets.	64-bit	TLB Invalidate by VA, All ASID, EL1
TLBI_VALE1	1	0	C8	C7	5	See individual bit resets.	64-bit	TLB Invalidate by VA, Last level, EL1
TLBI_VAALE1	1	0	C8	C7	7	See individual bit resets.	64-bit	TLB Invalidate by VA, All ASID, Last level, EL1
SYS_IMP_BPI	1	0	C15	C1	1	See individual bit resets.	64-bit	Branch Predictor Invalidation Operation
SYS_IMP_CDBGDCT	1	2	C15	C2	0	See individual bit resets.	64-bit	L1 Data Cache Tag Read Operation
SYS_IMP_CDBGICT	1	2	C15	C2	1	See individual bit resets.	64-bit	L1 Instruction Cache Tag Read Operation
SYS_IMP_CDBGTT	1	2	C15	C2	2	See individual bit resets.	64-bit	TLB Tag Read Operation
SYS_IMP_CLUSTERCDBGL2T	1	2	C15	C2	3	See individual bit resets.	64-bit	L2 Cache Tag Read Operation
SYS_IMP_CLUSTERCDBGL2DT	1	2	C15	C3	3	See individual bit resets.	64-bit	L2 Cache Duplicate L1 Tag Read Operation
SYS_IMP_CLUSTERCDBGLCUDT	1	2	C15	C3	4	See individual bit resets.	64-bit	LCU Duplicate L1 Tag Read Operation
SYS_IMP_CDBGDCD	1	2	C15	C4	0	See individual bit resets.	64-bit	L1 Data Cache Data Read Operation
SYS_IMP_CDBGICD	1	2	C15	C4	1	See individual bit resets.	64-bit	L1 Instruction Cache Data Read Operation
SYS_IMP_CDBGTD	1	2	C15	C4	2	See individual bit resets.	64-bit	TLB Data Read Operation
SYS_IMP_CLUSTERCDBGL2D	1	2	C15	C4	3	See individual bit resets.	64-bit	L2 Cache Data Read Operation

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
CFP_RCTX	1	3	C7	C3	4	See individual bit resets.	64-bit	Control Flow Prediction Restriction by Context
DVP_RCTX	1	3	C7	C3	5	See individual bit resets.	64-bit	Data Value Prediction Restriction by Context
CPP_RCTX	1	3	C7	C3	7	See individual bit resets.	64-bit	Cache Prefetch Prediction Restriction by Context
DC_ZVA	1	3	C7	C4	1	See individual bit resets.	64-bit	Data Cache Zero by VA
IC_IVAU	1	3	C7	C5	1	See individual bit resets.	64-bit	Instruction Cache line Invalidate by VA to PoU
DC_CVAC	1	3	C7	C10	1	See individual bit resets.	64-bit	Data or unified Cache line Clean by VA to PoC
DC_CVAU	1	3	C7	C11	1	See individual bit resets.	64-bit	Data or unified Cache line Clean by VA to PoU
DC_CVAP	1	3	C7	C12	1	See individual bit resets.	64-bit	Data or unified Cache line Clean by VA to PoP
DC_CVADP	1	3	C7	C13	1	See individual bit resets.	64-bit	Data or unified Cache line Clean by VA to PoDP
DC_CIVAC	1	3	C7	C14	1	See individual bit resets.	64-bit	Data or unified Cache line Clean and Invalidate by VA to PoC
AT_S1E2R	1	4	C7	C8	0	See individual bit resets.	64-bit	Address Translate Stage 1 EL2 Read
AT_S1E2W	1	4	C7	C8	1	See individual bit resets.	64-bit	Address Translate Stage 1 EL2 Write
AT_S12E1R	1	4	C7	C8	4	See individual bit resets.	64-bit	Address Translate Stages 1 and 2 EL1 Read
AT_S12E1W	1	4	C7	C8	5	See individual bit resets.	64-bit	Address Translate Stages 1 and 2 EL1 Write
AT_S12E0R	1	4	C7	C8	6	See individual bit resets.	64-bit	Address Translate Stages 1 and 2 EL0 Read
AT_S12E0W	1	4	C7	C8	7	See individual bit resets.	64-bit	Address Translate Stages 1 and 2 EL0 Write

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TLBI_IPAS2E1IS	1	4	C8	C0	1	See individual bit resets.	64-bit	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
TLBI_RIPAS2E1IS	1	4	C8	C0	2	See individual bit resets.	64-bit	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
TLBI_IPAS2LE1IS	1	4	C8	C0	5	See individual bit resets.	64-bit	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
TLBI_RIPAS2LE1IS	1	4	C8	C0	6	See individual bit resets.	64-bit	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
TLBI_ALLE2OS	1	4	C8	C1	0	See individual bit resets.	64-bit	TLB Invalidate All, EL2, Outer Shareable
TLBI_VAE2OS	1	4	C8	C1	1	See individual bit resets.	64-bit	TLB Invalidate by VA, EL2, Outer Shareable
TLBI_ALLE1OS	1	4	C8	C1	4	See individual bit resets.	64-bit	TLB Invalidate All, EL1, Outer Shareable
TLBI_VALE2OS	1	4	C8	C1	5	See individual bit resets.	64-bit	TLB Invalidate by VA, Last level, EL2, Outer Shareable
TLBI_VMALLS12E1OS	1	4	C8	C1	6	See individual bit resets.	64-bit	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Outer Shareable
TLBI_RVAE2IS	1	4	C8	C2	1	See individual bit resets.	64-bit	TLB Range Invalidate by VA, EL2, Inner Shareable
TLBI_RVALE2IS	1	4	C8	C2	5	See individual bit resets.	64-bit	TLB Range Invalidate by VA, Last level, EL2, Inner Shareable
TLBI_ALLE2IS	1	4	C8	C3	0	See individual bit resets.	64-bit	TLB Invalidate All, EL2, Inner Shareable
TLBI_VAE2IS	1	4	C8	C3	1	See individual bit resets.	64-bit	TLB Invalidate by VA, EL2, Inner Shareable
TLBI_ALLE1IS	1	4	C8	C3	4	See individual bit resets.	64-bit	TLB Invalidate All, EL1, Inner Shareable
TLBI_VALE2IS	1	4	C8	C3	5	See individual bit resets.	64-bit	TLB Invalidate by VA, Last level, EL2, Inner Shareable
TLBI_VMALLS12E1IS	1	4	C8	C3	6	See individual bit resets.	64-bit	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TLBI_IPAS2E1OS	1	4	C8	C4	0	See individual bit resets.	64-bit	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
TLBI_IPAS2E1	1	4	C8	C4	1	See individual bit resets.	64-bit	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
TLBI_RIPAS2E1	1	4	C8	C4	2	See individual bit resets.	64-bit	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1
TLBI_RIPAS2E1OS	1	4	C8	C4	3	See individual bit resets.	64-bit	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
TLBI_IPAS2LE1OS	1	4	C8	C4	4	See individual bit resets.	64-bit	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
TLBI_IPAS2LE1	1	4	C8	C4	5	See individual bit resets.	64-bit	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
TLBI_RIPAS2LE1	1	4	C8	C4	6	See individual bit resets.	64-bit	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
TLBI_RIPAS2LE1OS	1	4	C8	C4	7	See individual bit resets.	64-bit	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
TLBI_RVAE2OS	1	4	C8	C5	1	See individual bit resets.	64-bit	TLB Range Invalidate by VA, EL2, Outer Shareable
TLBI_RVALE2OS	1	4	C8	C5	5	See individual bit resets.	64-bit	TLB Range Invalidate by VA, Last level, EL2, Outer Shareable
TLBI_RVAE2	1	4	C8	C6	1	See individual bit resets.	64-bit	TLB Range Invalidate by VA, EL2
TLBI_RVALE2	1	4	C8	C6	5	See individual bit resets.	64-bit	TLB Range Invalidate by VA, Last level, EL2
TLBI_ALLE2	1	4	C8	C7	0	See individual bit resets.	64-bit	TLB Invalidate All, EL2
TLBI_VAE2	1	4	C8	C7	1	See individual bit resets.	64-bit	TLB Invalidate by VA, EL2
TLBI_ALLE1	1	4	C8	C7	4	See individual bit resets.	64-bit	TLB Invalidate All, EL1
TLBI_VALE2	1	4	C8	C7	5	See individual bit resets.	64-bit	TLB Invalidate by VA, Last level, EL2

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TLBI_VMALLS12E1	1	4	C8	C7	6	See individual bit resets.	64-bit	TLB Invalidate by VMID, All at Stage 1 and 2, EL1

A.1.5 AArch64 Debug registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Debug 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-5: 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
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
OSDTRTX_EL1	2	0	C0	C3	2	See individual bit resets.	64-bit	OS Lock Data Transfer Register, Transmit
OSECCR_EL1	2	0	C0	C6	2	See individual bit resets.	64-bit	OS Lock Exception Catch Control Register
DBGBVR<n>_EL1	2	0	C0	m[3:0]	4	See individual bit resets.	64-bit	Debug Breakpoint Value Registers
DBGBCR<n>_EL1	2	0	C0	m[3:0]	5	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
DBGWVR<n>_EL1	2	0	C0	m[3:0]	6	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
DBGWCR<n>_EL1	2	0	C0	m[3:0]	7	See individual bit resets.	64-bit	Debug Watchpoint Control Registers
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_ELO	2	3	C0	C1	0	See individual bit resets.	64-bit	Monitor DCC Status Register
DBGDTR_ELO	2	3	C0	C4	0	See individual bit resets.	64-bit	Debug Data Transfer Register, half-duplex
DBGDTRRX_ELO	2	3	C0	C5	0	See individual bit resets.	64-bit	Debug Data Transfer Register, Receive
DBGDTRTX_ELO	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)

A.1.6 AArch64 GIC system registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** GIC system 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-6: 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_EOIRO_EL1	3	0	C12	C8	1	See individual bit resets.	64-bit	Interrupt Controller End Of Interrupt Register 0
ICV_EOIRO_EL1	3	0	C12	C8	1	See individual bit resets.	64-bit	Interrupt Controller Virtual End Of Interrupt Register 0
ICC_AP0R<n>_EL1	3	0	C12	C8	1:m[1:0]	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R<n>_EL1	3	0	C12	C8	1:m[1:0]	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_HPPIRO_EL1	3	0	C12	C8	2	See individual bit resets.	64-bit	Interrupt Controller Highest Priority Pending Interrupt Register 0
ICV_HPPIRO_EL1	3	0	C12	C8	2	See individual bit resets.	64-bit	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
ICV_BPRO_EL1	3	0	C12	C8	3	See individual bit resets.	64-bit	Interrupt Controller Virtual Binary Point Register 0
ICC_AP1R<n>_EL1	3	0	C12	C9	0:m[1:0]	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R<n>_EL1	3	0	C12	C9	0:m[1:0]	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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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_LR<n>_EL2	3	4	C12	110:m[3]	m[2:0]	See individual bit resets.	64-bit	Interrupt Controller List Registers
ICH_AP0R<n>_EL2	3	4	C12	C8	0:m[1:0]	See individual bit resets.	64-bit	Interrupt Controller Hyp Active Priorities Group 0 Registers
ICH_AP1R<n>_EL2	3	4	C12	C9	0:m[1:0]	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

A.1.7 AArch64 RAS registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-7: 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
ERXMISCO_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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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 SErrors Exception Syndrome Register
VDISR_EL2	3	4	C12	C1	1	See individual bit resets.	64-bit	Virtual Deferred Interrupt Status Register

A.1.8 AArch64 Trace unit registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Trace unit 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-8: Trace unit registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCSEQEVR<n>	2	1	C0	00:m[1:0]	4	See individual bit resets.	64-bit	Sequencer State Transition Control Register <n>
TRCCNTRLDVR<n>	2	1	C0	00:m[1:0]	5	See individual bit resets.	64-bit	Counter Reload Value Register <n>
TRCCNTCTLR<n>	2	1	C0	01:m[1:0]	5	See individual bit resets.	64-bit	Counter Control Register <n>
TRCCNTVR<n>	2	1	C0	10:m[1:0]	5	See individual bit resets.	64-bit	Counter Value Register <n>
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
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
TRCIDR9	2	1	C0	C1	6	See individual bit resets.	64-bit	ID Register 9

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCVISSCTLR	2	1	C0	C2	2	See individual bit resets.	64-bit	ViewInst Start/Stop Control Register
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
TRCIDR12	2	1	C0	C4	6	See individual bit resets.	64-bit	ID Register 12
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	Auxillary Control Register
TRCSEQRSTEV	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
TRCEVENTCTLOR	2	1	C0	C8	0	See individual bit resets.	64-bit	Event Control 0 Register
TRCVDCTLR	2	1	C0	C8	2	See individual bit resets.	64-bit	ViewData Main Control Register
TRCEXTINSEL	2	1	C0	C8	4	See individual bit resets.	64-bit	External Input Select Register
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
TRCVDSACCTLR	2	1	C0	C9	2	See individual bit resets.	64-bit	ViewData Include/Exclude Single Address Comparator Control Register
TRCIDR1	2	1	C0	C9	7	See individual bit resets.	64-bit	ID Register 1
TRCVDARCCTLR	2	1	C0	C10	2	See individual bit resets.	64-bit	ViewData Include/Exclude Address Range Comparator Control Register
TRCIDR2	2	1	C0	C10	7	See individual bit resets.	64-bit	ID Register 2
TRCSTALLCTLR	2	1	C0	C11	0	See individual bit resets.	64-bit	Stall Control Register
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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
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
TRCSSCCR<n>	2	1	C1	0:m[2:0]	2	See individual bit resets.	64-bit	Single-shot Comparator Control Register <n>
TRCSSCSR<n>	2	1	C1	1:m[2:0]	2	See individual bit resets.	64-bit	Single-shot Comparator Control Status Register <n>
TRCOSLAR	2	1	C1	C0	4	See individual bit resets.	64-bit	Trace OS Lock Access Register
TRCOSLSR	2	1	C1	C1	4	See individual bit resets.	64-bit	Trace OS Lock Status Register
TRCRSCTLR<n>	2	1	C1	m[3:0]	00:m[4]	See individual bit resets.	64-bit	Resource Selection Control Register <n>
TRCDVCVR<n>	2	1	C2	m[1:0]:00	10:m[2]	See individual bit resets.	64-bit	Data Value Comparator Value Register <n>
TRCDVCMR<n>	2	1	C2	m[1:0]:00	11:m[2]	See individual bit resets.	64-bit	Data Value Comparator Mask Register <n>
TRCACVR<n>	2	1	C2	m[2:0]:0	00:m[3]	See individual bit resets.	64-bit	Address Comparator Value Register <n>
TRCACATR<n>	2	1	C2	m[2:0]:0	01:m[3]	See individual bit resets.	64-bit	Address Comparator Access Type 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
TRCCIDCVR<n>	2	1	C3	m[2:0]:0	0	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>
TRCVMIDCVR<n>	2	1	C3	m[2:0]:0	1	See individual bit resets.	64-bit	Virtual Context Identifier Comparator Value Register <n>
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

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
TRCDEVARCH	2	1	C7	C15	6	See individual bit resets.	64-bit	Device Architecture Register

A.1.9 AArch64 Special-purpose registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Special-purpose 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-9: 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_ELO	3	0	C4	C1	0	See individual bit resets.	64-bit	Stack Pointer (ELO)
DPSR_ELO	3	3	C4	C5	0	See individual bit resets.	64-bit	Debug Saved Program Status Register
DLR_ELO	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)

A.1.10 AArch64 Generic Timer registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** Generic Timer 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-10: 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
CNTRQ_ELO	3	3	C14	C0	0	See individual bit resets.	64-bit	Counter-timer Frequency Register
CNTPCT_ELO	3	3	C14	C0	1	See individual bit resets.	64-bit	Counter-timer Physical Count Register
CNTVCT_ELO	3	3	C14	C0	2	See individual bit resets.	64-bit	Counter-timer Virtual Count Register
CNTP_TVAL_ELO	3	3	C14	C2	0	See individual bit resets.	64-bit	Counter-timer Physical Timer TimerValue Register
CNTP_CTL_ELO	3	3	C14	C2	1	See individual bit resets.	64-bit	Counter-timer Physical Timer Control Register
CNTP_CVAL_ELO	3	3	C14	C2	2	See individual bit resets.	64-bit	Counter-timer Physical Timer CompareValue Register
CNTV_TVAL_ELO	3	3	C14	C3	0	See individual bit resets.	64-bit	Counter-timer Virtual Timer TimerValue Register
CNTV_CTL_ELO	3	3	C14	C3	1	See individual bit resets.	64-bit	Counter-timer Virtual Timer Control Register
CNTV_CVAL_ELO	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
CNTHCTL_EL2	3	4	C14	C1	0	See individual bit resets.	64-bit	Counter-timer Hypervisor Control Register
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)

A.1.11 AArch64 Other system control registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the [Arm® Architecture Reference Manual for A-profile architecture](#).

Table A-11: Other system control registers summary

Name	Op0	Op1	CRn	CRm	Op2	Reset	Width	Description
SCTLR_EL2	3	4	C1	C0	0	See individual bit resets.	64-bit	System Control Register (EL2)
HSTR_EL2	3	4	C1	C1	3	See individual bit resets.	64-bit	Hypervisor System Trap Register

A.2 AArch64 register descriptions

This section includes the register descriptions for all the AArch64 registers in the Cortex®-R82 processor.

A.2.1 AArch64 Identification register description

This section includes the register descriptions for all Identification registers in the Cortex®-R82 processor.

A.2.1.1 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

AArch64 register MIDR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.1.17 MIDR_EL1, Main ID Register](#) on page 1658 bits [31:0].

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0100	0001	0011	1111	1101	0001	0101	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-1: AARCH64_MIDR_EL1 bit assignments

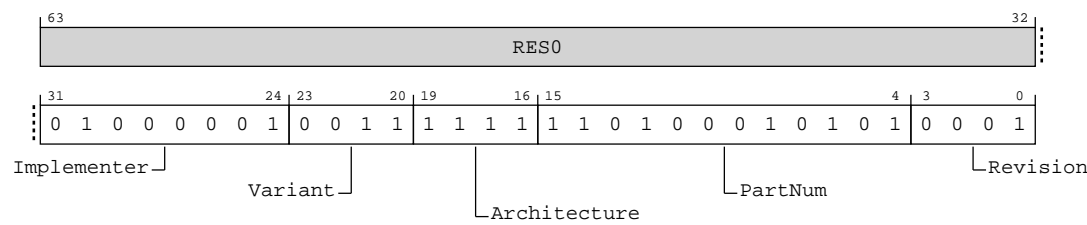


Table A-12: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	Implementer	The Implementer code. This field must hold an implementer code that has been assigned by Arm. 0b01000001 Arm Limited	0x41
[23:20]	Variant	The major product revision variant number. 0b0011 r3.	0b0011
[19:16]	Architecture	Architecture version. Defined values are: 0b1111 Architectural features are individually identified in the ID_* registers. All other values are reserved.	0b1111
[15:4]	PartNum	The primary part number for the device. 0b110100010101 Cortex-R82.	0xD15
[3:0]	Revision	The minor product revision variant number. 0b0001 p1.	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 HCR_EL2.TGE == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    AArch64.SystemAccessTrap(EL1, 0x18);
```

```
elseif PSTATE.EL == EL1 then
    X[t, 64] = VPIDR_EL2;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MIDR_EL1;
```

A.2.1.2 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

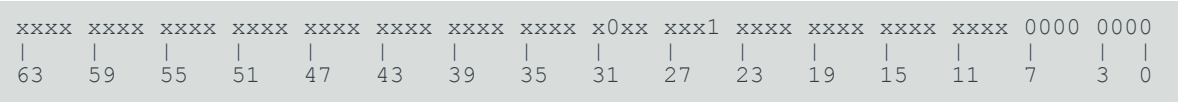
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-2: AARCH64_MPIDR_EL1 bit assignments

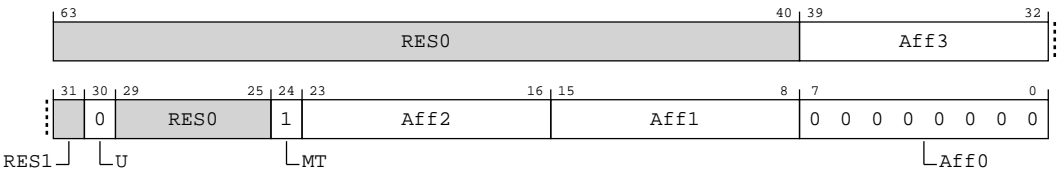


Table A-14: MPIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[39:32]	Aff3	Affinity level 3. See the description of Aff0 for more information. Value read from the CFGMPIDRAFF3 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. Value read from the CFGMPIDRAFF2 configuration pins.	8 {x}
[15:8]	Aff1	Affinity level 1. See the description of Aff0 for more information. 0b00000000 Core 0. 0b00000001 Core 1. 0b00000010 Core 2. 0b00000011 Core 3. 0b00000100 Core 4. 0b00000101 Core 5. 0b00000110 Core 6. 0b00000111 Core 7.	8 {x}
[7:0]	Aff0	Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the AArch64-MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole. 0b00000000 Single 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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    X[t, 64] = VMPIDR_EL2;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MPIDR_EL1;
```

A.2.1.3 REVIDR_EL1, Revision ID Register

Provides implementation-specific minor revision information.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-3: AARCH64_REVIDR_EL1 bit assignments

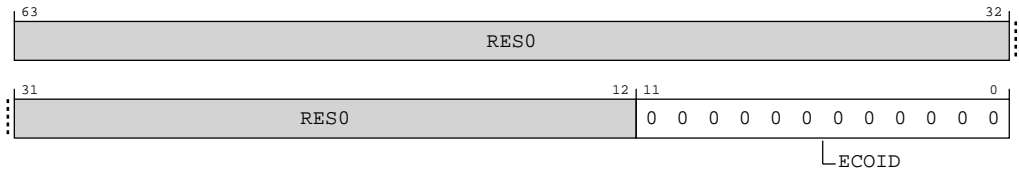


Table A-16: REVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	RES0
[11:0]	ECOID	Contains information about the ECOs applied to this processor. 0b000000000000 No ECO applied.	0x000

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
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = REVIDR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = REVIDR_EL1;
```

A.2.1.4 ID_PFR0_EL1, AArch32 Processor Feature Register 0

Gives top-level information about the instruction sets supported by the PE in AArch32 state.

Must be interpreted with AArch64-ID_PFR1_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

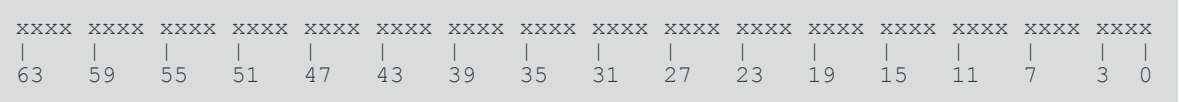
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-4: AARCH64_ID_PFR0_EL1 bit assignments

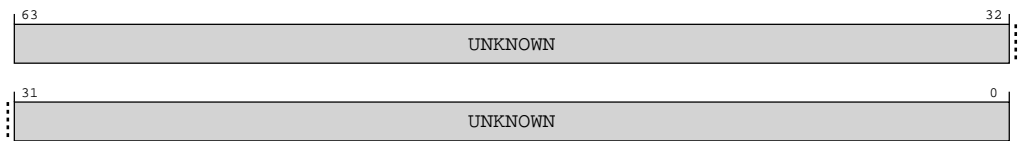


Table A-18: ID_PFR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_PFR0_EL1

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

Accessibility

MRS <Xt>, ID_PFR0_EL1

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

```
if HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = ID_PFR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_PFR0_EL1;
```

A.2.1.5 ID_PFR1_EL1, AArch32 Processor Feature Register 1

Gives information about the AArch32 programmers' model.

Must be interpreted with AArch64-ID_PFR0_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-5: AARCH64_ID_PFR1_EL1 bit assignments

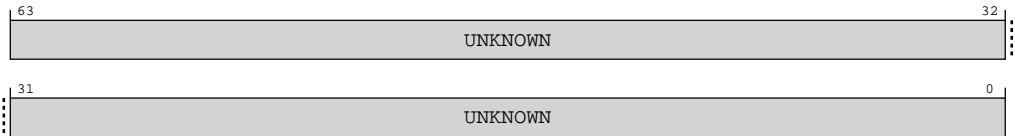


Table A-20: ID_PFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_PFR1_EL1

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

Accessibility

MRS <Xt>, ID_PFR1_EL1

```

if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_PFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_PFR1_EL1;

```

A.2.1.6 ID_DFR0_EL1, AArch32 Debug Feature Register 0

Provides top level information about the debug system in AArch32 state.

Must be interpreted with the Main ID Register, AArch64-MIDR_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes**Width**

64

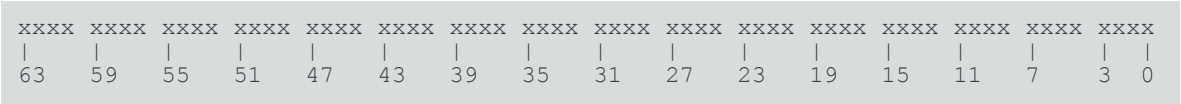
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-6: AArch64_ID_DFR0_EL1 bit assignments

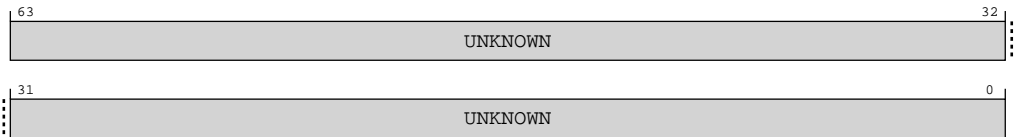


Table A-22: ID_DFR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_DFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0001	0b010

Accessibility

MRS <Xt>, ID_DFR0_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_DFR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_DFR0_EL1;
```

A.2.1.7 ID_AFR0_EL1, AArch32 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch32 state.

Must be interpreted with the Main ID Register, AArch64-MIDR_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

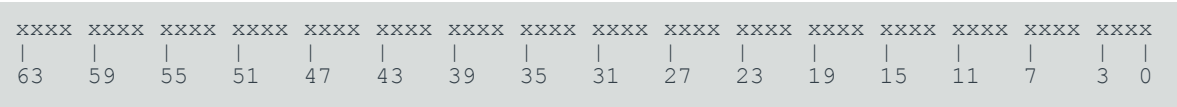
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-7: AARCH64_ID_AFR0_EL1 bit assignments

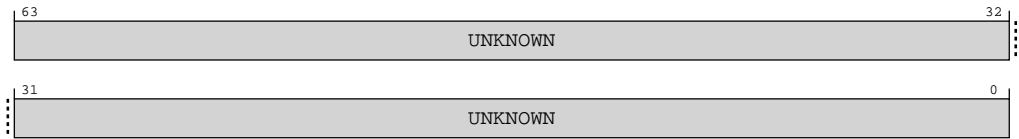


Table A-24: ID_AFR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0001	0b011

Accessibility

MRS <Xt>, ID_AFR0_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AFR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AFR0_EL1;
```

A.2.1.8 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch32 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

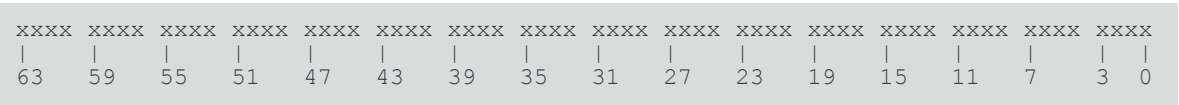
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-8: AARCH64_ID_MMFR0_EL1 bit assignments

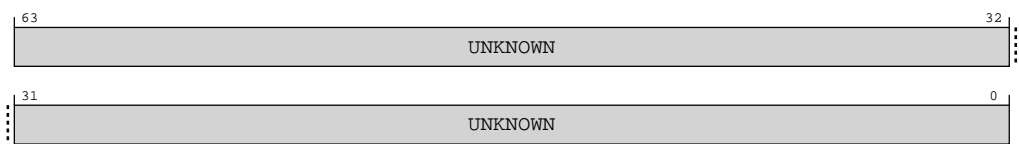


Table A-26: ID_MMFR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_MMFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0001	0b100

Accessibility

MRS <Xt>, ID_MMFR0_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_MMFR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_MMFR0_EL1;
```

A.2.1.9 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch32 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

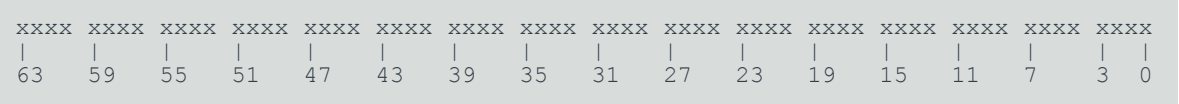
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-9: AARCH64_ID_MMFR1_EL1 bit assignments

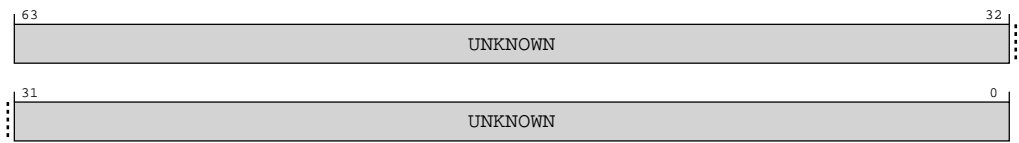


Table A-28: ID_MMFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_MMFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0001	0b101

Accessibility

MRS <Xt>, ID_MMFR1_EL1

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



```
if HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = ID_MMFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_MMFR1_EL1;
```

A.2.1.10 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch32 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-10: AARCH64_ID_MMFR2_EL1 bit assignments

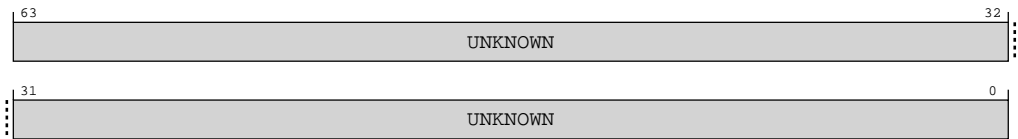


Table A-30: ID_MMFR2_EL1 bit descriptions

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

Access
MRS <Xt>, ID_MMFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0001	0b110

Accessibility
MRS <Xt>, ID_MMFR2_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_MMFR2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_MMFR2_EL1;
```

A.2.1.11 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3

Provides information about the implemented memory model and memory management support in AArch32 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
```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-11: AARCH64_ID_MMFR3_EL1 bit assignments

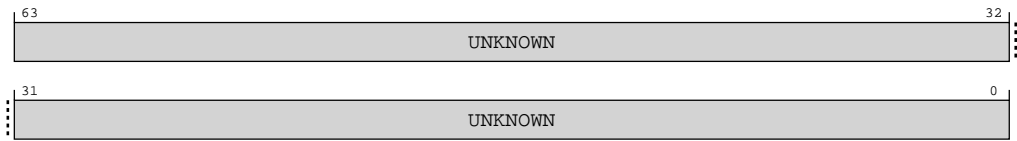


Table A-32: ID_MMFR3_EL1 bit descriptions

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

Access

MRS <Xt>, ID_MMFR3_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0001	0b111

Accessibility

MRS <Xt>, ID_MMFR3_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_MMFR3_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_MMFR3_EL1;
```

A.2.1.12 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

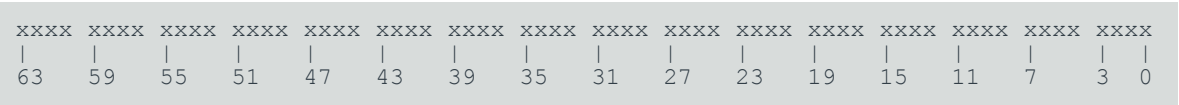
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-12: AARCH64_ID_ISAR0_EL1 bit assignments

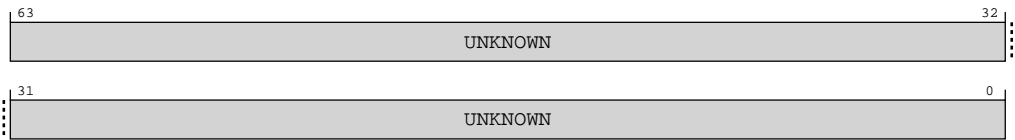


Table A-34: ID_ISAR0_EL1 bit descriptions

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

Access

MRS <Xt>, ID_ISARO_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b000

Accessibility

MRS <Xt>, ID_ISARO_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_ISARO_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISARO_EL1;
```

A.2.1.13 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISARO_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-13: AARCH64_ID_ISAR1_EL1 bit assignments

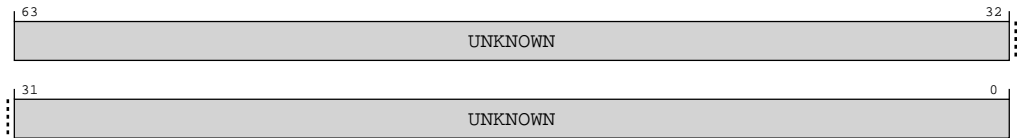


Table A-36: ID_ISAR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_ISAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b001

Accessibility

MRS <Xt>, ID_ISAR1_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_ISAR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISAR1_EL1;
```

A.2.1.14 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

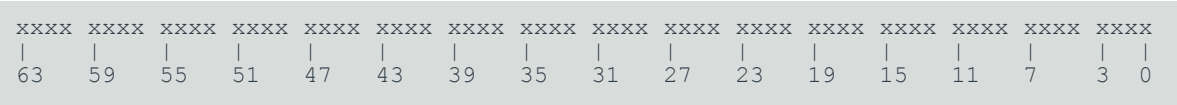
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-14: AARCH64_ID_ISAR2_EL1 bit assignments

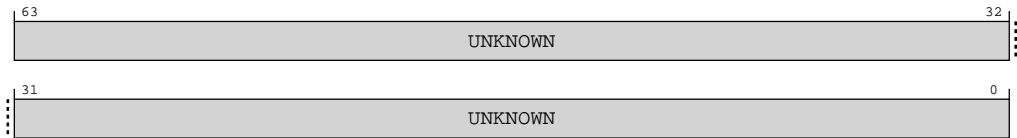


Table A-38: ID_ISAR2_EL1 bit descriptions

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

Access

MRS <Xt>, ID_ISAR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b010

Accessibility

MRS <Xt>, ID_ISAR2_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_ISAR2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISAR2_EL1;
```

A.2.1.15 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* 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-15: AARCH64_ID_ISAR3_EL1 bit assignments

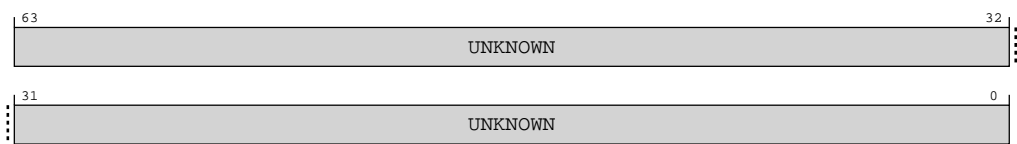


Table A-40: ID_ISAR3_EL1 bit descriptions

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

Access

MRS <Xt>, ID_ISAR3_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b011

Accessibility

MRS <Xt>, ID_ISAR3_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_ISAR3_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISAR3_EL1;
```

A.2.1.16 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

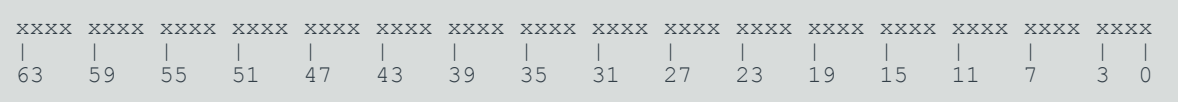
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-16: AARCH64_ID_ISAR4_EL1 bit assignments

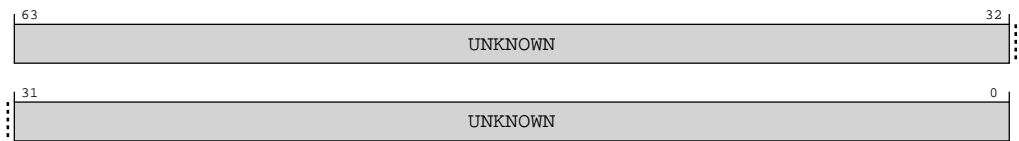


Table A-42: ID_ISAR4_EL1 bit descriptions

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

Access

MRS <Xt>, ID_ISAR4_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b100

Accessibility

MRS <Xt>, ID_ISAR4_EL1

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

```
if HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = ID_ISAR4_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISAR4_EL1;
```

A.2.1.17 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, and AArch64-ID_ISAR4_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

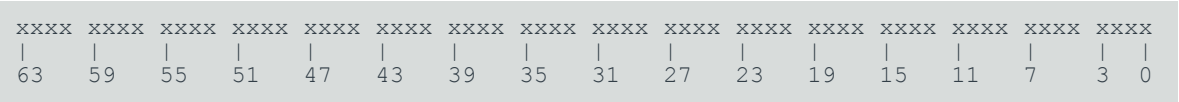
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-17: AARCH64_ID_ISAR5_EL1 bit assignments

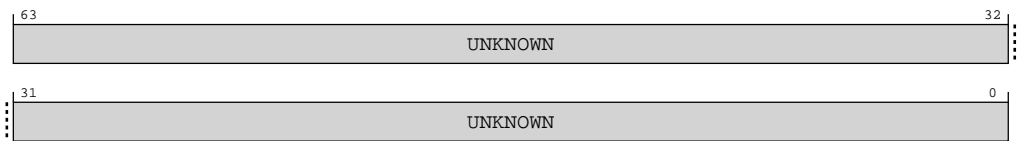


Table A-44: ID_ISAR5_EL1 bit descriptions

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

Access

MRS <Xt>, ID_ISAR5_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b101

Accessibility

MRS <Xt>, ID_ISAR5_EL1

```

if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_ISAR5_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISAR5_EL1;

```

A.2.1.18 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4

Provides information about the implemented memory model and memory management support in AArch32 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

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

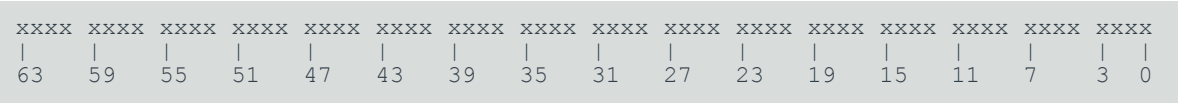
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-18: AARCH64_ID_MMFR4_EL1 bit assignments

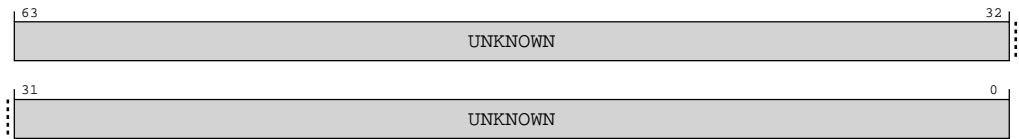


Table A-46: ID_MMFR4_EL1 bit descriptions

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

Access

MRS <Xt>, ID_MMFR4_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b110

Accessibility

MRS <Xt>, ID_MMFR4_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_MMFR4_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_MMFR4_EL1;
```

A.2.1.19 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1 and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations



Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	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-19: AARCH64_ID_ISAR6_EL1 bit assignments

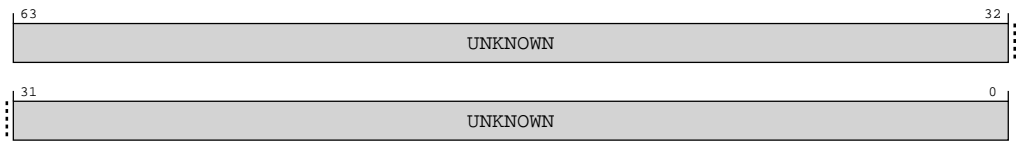


Table A-48: ID_ISAR6_EL1 bit descriptions

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

Access

MRS <Xt>, ID_ISAR6_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b111

Accessibility

MRS <Xt>, ID_ISAR6_EL1

```

if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_ISAR6_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISAR6_EL1;

```

A.2.1.20 MVFR0_EL1, AArch32 Media and VFP Feature Register 0

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR1_EL1 and AArch64-MVFR2_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

Attributes**Width**

64

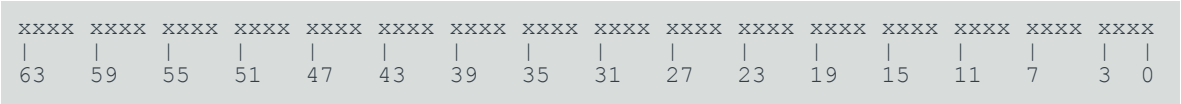
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-20: AARCH64_MVFR0_EL1 bit assignments

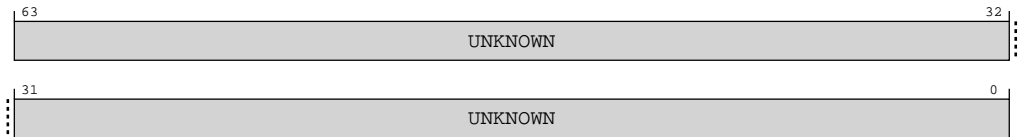


Table A-50: MVFR0_EL1 bit descriptions

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

Access

MRS <Xt>, MVFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b000

Accessibility

MRS <Xt>, MVFR0_EL1

```
if PSTATE.EL == EL0 then
  if HCR_EL2.TGE == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
  if HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    X[t, 64] = MVFR0_EL1;
elseif PSTATE.EL == EL2 then
  X[t, 64] = MVFR0_EL1;
```


A.2.1.21 MVFR1_EL1, AArch32 Media and VFP Feature Register 1

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR0_EL1 and AArch64-MVFR2_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

Attributes

Width

64

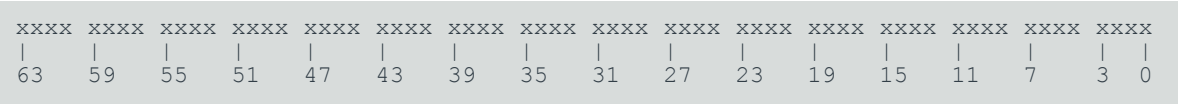
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-21: AARCH64_MVFR1_EL1 bit assignments

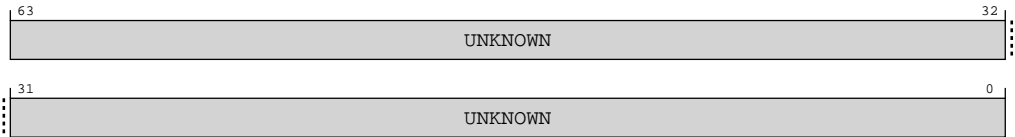


Table A-52: MVFR1_EL1 bit descriptions

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

Access

MRS <Xt>, MVFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b001

Accessibility

MRS <Xt>, MVFR1_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MVFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MVFR1_EL1;
```

A.2.1.22 MVFR2_EL1, AArch32 Media and VFP Feature Register 2

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR0_EL1 and AArch64-MVFR1_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-22: AARCH64_MVFR2_EL1 bit assignments

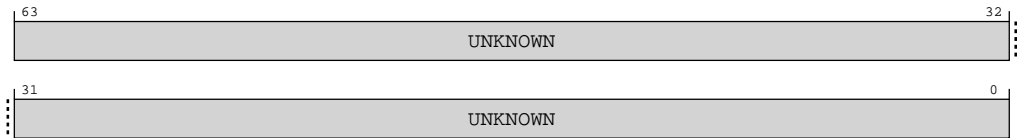


Table A-54: MVFR2_EL1 bit descriptions

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

Access

MRS <Xt>, MVFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b010

Accessibility

MRS <Xt>, MVFR2_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MVFR2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MVFR2_EL1;
```

A.2.1.23 ID_PFR2_EL1, AArch32 Processor Feature Register 2

Gives information about the AArch32 programmers' model.

Must be interpreted with AArch64-ID_PFR0_EL1 and AArch64-ID_PFR1_EL1.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* 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

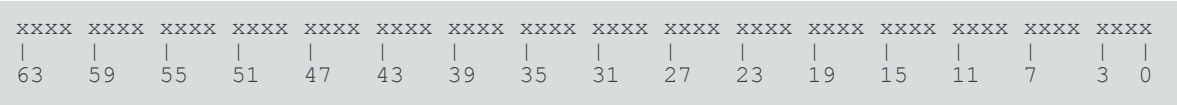
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-23: AARCH64_ID_PFR2_EL1 bit assignments

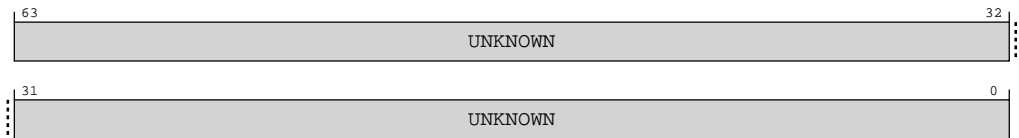


Table A-56: ID_PFR2_EL1 bit descriptions

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

Access

MRS <Xt>, ID_PFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b100

Accessibility

MRS <Xt>, ID_PFR2_EL1


```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if (!IsZero(ID_PFR2_EL1) || boolean IMPLEMENTATION_DEFINED "ID_PFR2_EL1 trapped
by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_PFR2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_PFR2_EL1;
```

A.2.1.24 ID_DFR1_EL1, Debug Feature Register 1

Provides top level information about the debug system in AArch32.

For general information about the interpretation of the ID registers see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations



Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group


Identification registers

Access type

See bit descriptions

Reset value

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



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-24: AARCH64_ID_DFR1_EL1 bit assignments

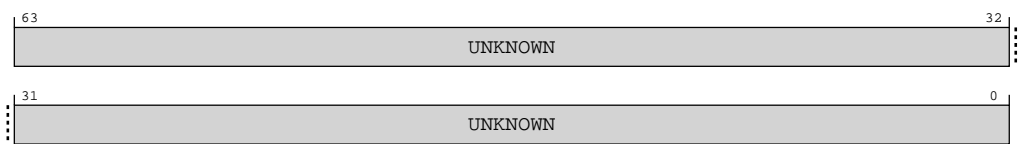


Table A-58: ID_DFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_DFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b101

Accessibility

MRS <Xt>, ID_DFR1_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_DFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_DFR1_EL1;
```

A.2.1.25 ID_MMFR5_EL1, AArch32 Memory Model Feature Register 5

Provides information about the implemented memory model and memory management support in AArch32 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

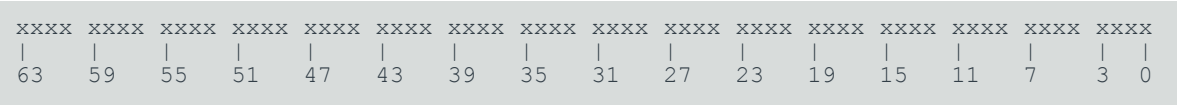
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-25: AARCH64_ID_MMFR5_EL1 bit assignments

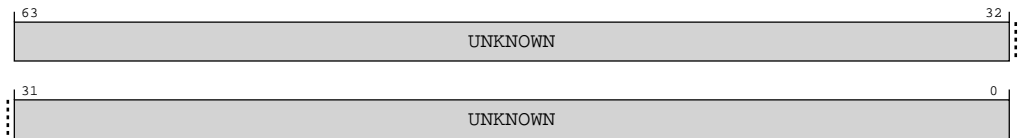


Table A-60: ID_MMFR5_EL1 bit descriptions

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

Access

MRS <Xt>, ID_MMFR5_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b110

Accessibility

MRS <Xt>, ID_MMFR5_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_MMFR5_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_MMFR5_EL1;
```

A.2.1.26 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	0001	xxxx	0001	xxxx	xxxx	0001	xxxx	0010	xxxx	xxxx	xxxx	0000	0001	0001	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-26: AARCH64_ID_AA64PFR0_EL1 bit assignments

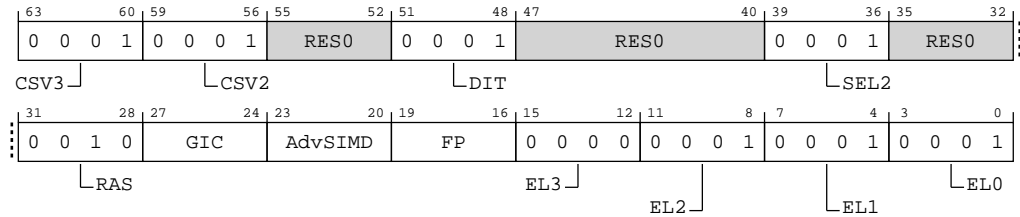


Table A-62: ID_AA64PFR0_EL1 bit descriptions

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: 0b0001 FEAT_CSV2 is implemented, but FEAT_CSV2_2 and FEAT_CSV2_3 are not implemented. AArch64-ID_AA64PFR1_EL1.CSV2_frac determines whether either or both of FEAT_CSV2_1p1 or FEAT_CSV2_1p2 are implemented.	0b0001
[55:52]	RES0	Reserved	RES0
[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
[47:40]	RES0	Reserved	RES0
[39:36]	SEL2	Secure EL2. Defined values are: 0b0001 Secure EL2 is implemented.	0b0001
[35:32]	RES0	Reserved	RES0
[31:28]	RAS	RAS Extension version. 0b0010 FEAT_RASv1p1 implemented. It also adds support for: <ul style="list-style-type: none"> Additional ERXMISC<m>_EL1 System registers. Additional System registers AArch64-ERXPFPGCDN_EL1, AArch64-ERXPFGCTL_EL1, and AArch64-ERXPFGF_EL1, and the AArch64-HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension. Error records accessed through System registers conform to RAS System Architecture v1.1, which includes simplifications to ext-ERR<n>STATUS and support for the optional RAS Timestamp and RAS Common Fault Injection Model Extensions.	0b0010

Bits	Name	Description	Reset
[27:24]	GIC	<p>When GICCDISABLE == 0 System register GIC interface support.</p> <p>0001 GIC CPU interface enabled. System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported.</p> <p>Otherwise System register GIC interface support.</p> <p>0000 GIC CPU interface disabled. System register interface to any external GIC not supported.</p>	xxxx
[23:20]	AdvSIMD	<p>When NEON_FPm > 0 Advanced SIMD.</p> <p>0001 Advanced SIMD is implemented that includes the FEAT_FP16 extension.</p> <p>Otherwise Advanced SIMD.</p> <p>1111 Advanced SIMD is not implemented.</p>	xxxx
[19:16]	FP	<p>When NEON_FPm > 0 Floating-point.</p> <p>0001 Floating-point is implemented that includes the FEAT_FP16 extension.</p> <p>Otherwise Floating-point.</p> <p>1111 Floating-point is not implemented.</p>	xxxx
[15:12]	EL3	<p>EL3 Exception level handling.</p> <p>0b0000 EL3 is not implemented.</p>	0b0000
[11:8]	EL2	<p>EL2 Exception level handling.</p> <p>0b0001 EL2 can be executed in AArch64 state only.</p>	0b0001
[7:4]	EL1	<p>EL1 Exception level handling.</p> <p>0b0001 EL1 can be executed in AArch64 state only.</p>	0b0001
[3:0]	ELO	<p>ELO Exception level handling.</p> <p>0b0001 ELO can be executed in AArch64 state only.</p>	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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR0_EL1;

```

A.2.1.27 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0010	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_ID_AA64PFR1_EL1 bit assignments

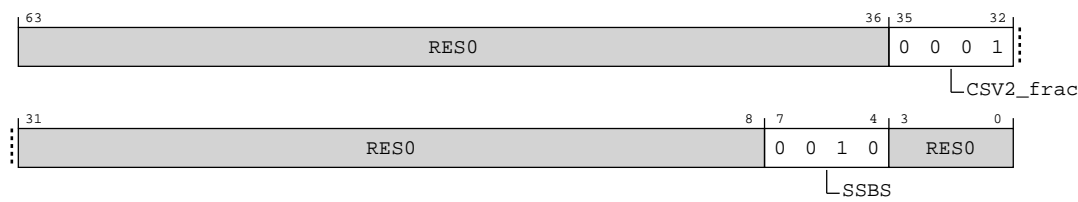


Table A-64: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:36]	RES0	Reserved	RES0
[35:32]	CSV2_frac	CSV2 fractional field. Defined values are: 0b0001 FEAT_CSV2_1p1 is implemented, but FEAT_CSV2_1p2 is not implemented.	0b0001
[31:8]	RES0	Reserved	RES0
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. 0b0010 AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe, and the MSR and MRS instructions to directly read and write the PSTATE.SSBS field.	0b0010
[3:0]	RES0	Reserved	RES0

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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64PFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64PFR1_EL1;
```

A.2.1.28 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

The external register ext-EDDFR gives information from this register.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	0001	1111	xxxx	0001	xxxx	0011	xxxx	0101	0101	0001	1001
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-28: AARCH64_ID_AA64DFR0_EL1 bit assignments

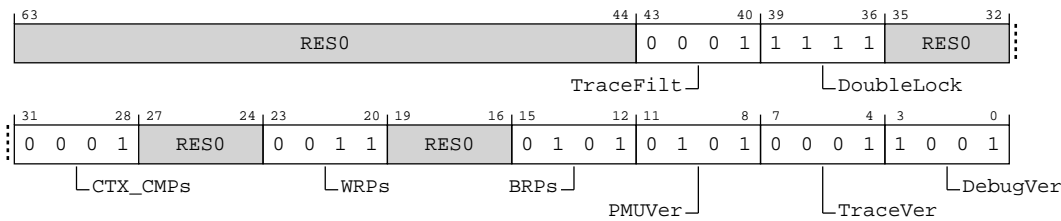


Table A-66: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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]	RES0	Reserved	RES0
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. 0b0001 2 context-aware breakpoints implemented. The value of this field is never greater than ID_AA64DFR0_EL1.BRPs.	0b0001
[27:24]	RES0	Reserved	RES0
[23:20]	WRPs	Number of watchpoints, minus 1. 0b0011 4 watchpoints implemented. The value of 0b0000 is reserved.	0b0011
[19:16]	RES0	Reserved	RES0
[15:12]	BRPs	Number of breakpoints, minus 1. 0b0101 6 breakpoints implemented. The value of 0b0000 is reserved.	0b0101
[11:8]	PMUVer	Performance Monitors Extension version. This field does not follow the standard ID scheme, but uses the alternative ID scheme described in <i>Alternative ID scheme used for the Performance Monitors Extension version</i> in the Arm® Architecture Reference Manual for A-profile architecture Defined values are: 0b0101 PMUv3 for Armv8.4. As 0b0100, and adds support for the AArch64-PMUVER_EL1 register.	0b0101
[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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64DFR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64DFR0_EL1;
```

A.2.1.29 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	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-29: AARCH64_ID_AA64DFR1_EL1 bit assignments

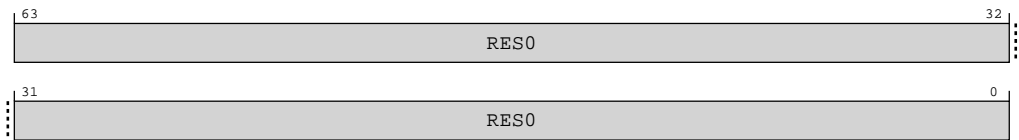


Table A-68: ID_AA64DFR1_EL1 bit descriptions

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

Access

MRS <Xt>, ID_AA64DFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0101	0b001

Accessibility

MRS <Xt>, ID_AA64DFR1_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64DFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64DFR1_EL1;
```

A.2.1.30 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

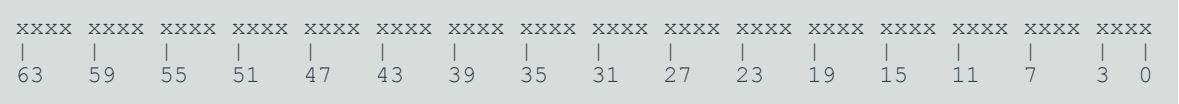
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-30: AARCH64_ID_AA64AFR0_EL1 bit assignments

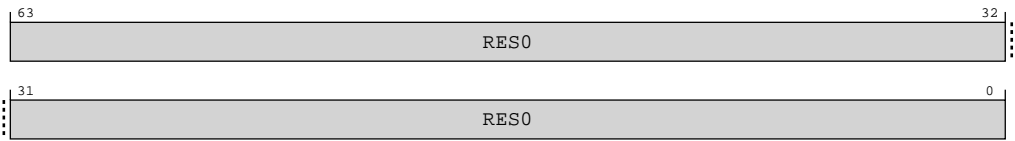


Table A-70: ID_AA64AFR0_EL1 bit descriptions

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 HCR_EL2.TGE == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
```

```
if HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = ID_AA64AFR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64AFR0_EL1;
```

A.2.1.31 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

Reserved for future expansion of information about the **IMPLEMENTATION DEFINED** features of the PE in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-31: AARCH64_ID_AA64AFR1_EL1 bit assignments

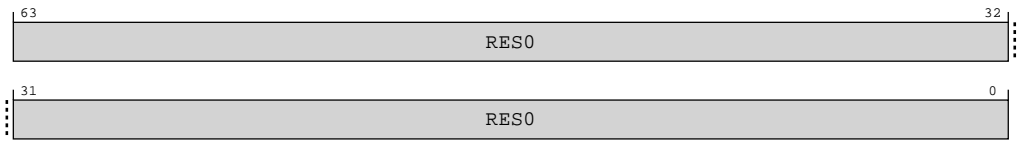


Table A-72: ID_AA64AFR1_EL1 bit descriptions

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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64AFR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64AFR1_EL1;
```

A.2.1.32 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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**



63 59 55 51 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_ID_AA64ISAR0_EL1 bit assignments

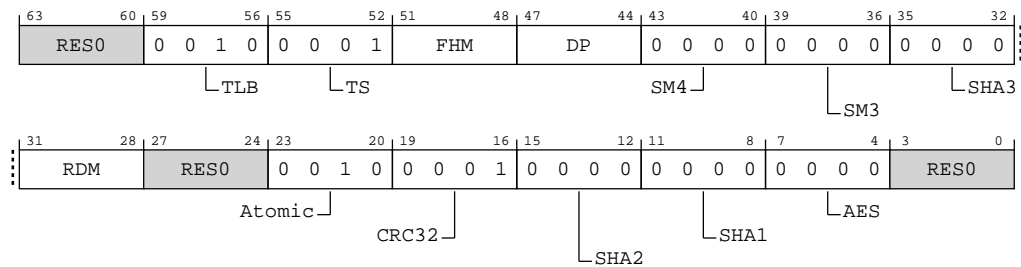


Table A-74: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RES0	Reserved	RES0
[59:56]	TLB	Indicates support for Outer Shareable and TLB range maintenance instructions. Defined values are: 0b0010 Outer Shareable and TLB range maintenance instructions are implemented.	0b0010
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: 0b0001 CFINV, RMIF, SETF16, and SETF8 instructions are implemented.	0b0001
[51:48]	FHM	When NEON_FPm > 0 Indicates whether FMLAL and FMLSL instructions are implemented. 0001 FMLAL and FMLSL instructions are implemented. Otherwise Indicates whether FMLAL and FMLSL instructions are implemented. 0000 FMLAL and FMLSL instructions are not implemented.	xxxx

Bits	Name	Description	Reset
[47:44]	DP	When NEON_FPm > 0 Dot Product instructions implemented. 0001 UDOT and SDOT instructions implemented. Otherwise Dot Product instructions implemented. 0000 No Dot Product instructions implemented.	xxxx
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: 0b0000 No SM4 instructions implemented.	0b0000
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: 0b0000 No SM3 instructions implemented.	0b0000
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: 0b0000 No SHA3 instructions implemented.	0b0000
[31:28]	RDM	When NEON_FPm > 0 SQRDMLAH and SQRDMLSH instructions implemented. 0001 SQRDMLAH and SQRDMLSH instructions implemented. Otherwise SQRDMLAH and SQRDMLSH instructions implemented. 0000 No SQRDMLAH and SQRDMLSH instructions implemented.	xxxx
[27:24]	RES0	Reserved	RES0
[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 No SHA2 instructions implemented.	0b0000
[11:8]	SHA1	Indicates support for SHA1 instructions in AArch64 state. Defined values are: 0b0000 No SHA1 instructions implemented.	0b0000

Bits	Name	Description	Reset
[7:4]	AES	Indicates support for AES instructions in AArch64 state. Defined values are: 0b0000 No AES instructions implemented.	0b0000
[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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR0_EL1;

```

A.2.1.33 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* 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	0001	xxxx	0001	0001	xxxx	0000	0000	0010	xxxx	xxxx	0000	0000	0010
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-33: AARCH64_ID_AA64ISAR1_EL1 bit assignments

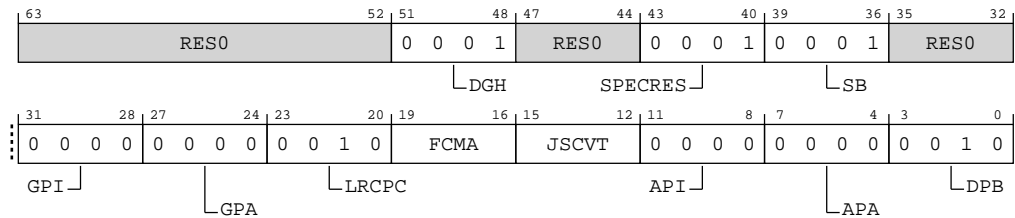


Table A-76: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:52]	RES0	Reserved	RES0
[51:48]	DGH	Indicates support for the Data Gathering Hint instruction. Defined values are: 0b0001 Data Gathering Hint is implemented.	0b0001
[47:44]	RES0	Reserved	RES0
[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]	RES0	Reserved	RES0
[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. 0b0000 Generic Authentication using the QARMA5 algorithm is not implemented.	0b0000

Bits	Name	Description	Reset
[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	When NEON_FPM > 0 Indicates support for complex number addition and multiplication, where numbers are stored in vectors. 0001 The FCMLA and FCADD instructions are implemented. Otherwise Indicates support for complex number addition and multiplication, where numbers are stored in vectors. 0000 The FCMLA and FCADD instructions are not implemented.	xxxx
[15:12]	JSCVT	When NEON_FPM > 0 Indicates support for javascript conversion from double precision floating point values to integers. 0001 The FJCVTZS instruction is implemented. Otherwise Indicates support for javascript conversion from double precision floating point values to integers. 0000 The FJCVTZS instruction is not implemented.	xxxx
[11:8]	API	Indicates whether an IMPLEMENTATION DEFINED algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: 0b0000 Address Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.	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. 0b0000 Address Authentication using the QARMA5 algorithm is not implemented.	0b0000
[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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64ISAR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR1_EL1;
```

A.2.1.34 ID_AA64ISAR2_EL1, AArch64 Instruction Set Attribute Register 2

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations



Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group


Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	xxxx	xxxx	0101	0001	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_ID_AA64ISAR2_EL1 bit assignments

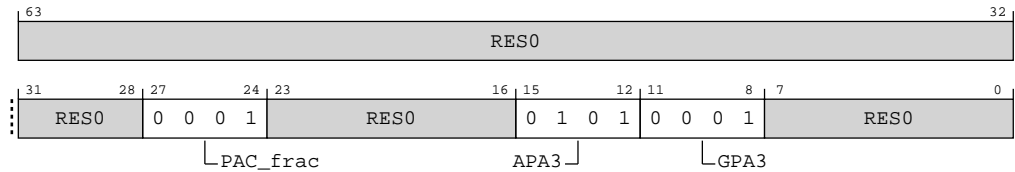


Table A-78: ID_AA64ISAR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:28]	RES0	Reserved	RES0
[27:24]	PAC_frac	Indicates whether the ConstPACField() function used as part of the PAC addition returns FALSE or TRUE. 0b0001 ConstPACField() returns TRUE.	0b0001
[23:16]	RES0	Reserved	RES0
[15:12]	APA3	Indicates whether the QARMA3 algorithm is implemented in the PE for address authentication in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. 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. 0b0001 Generic Authentication using the QARMA3 algorithm is implemented. This includes the PACGA instruction.	0b0001
[7:0]	RES0	Reserved	RES0

Access

MRS <Xt>, ID_AA64ISAR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b010

Accessibility

MRS <Xt>, ID_AA64ISAR2_EL1

```

if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if (!IsZero(ID_AA64ISAR2_EL1) || boolean IMPLEMENTATION_DEFINED
        "ID_AA64ISAR2_EL1_trapped by HCR_EL2.TID3") && 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;
```

A.2.1.35 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	1111	xxxx	xxxx	xxxx	xxxx	0000	0000	0001	xxxx	0001	0001	0010	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_ID_AA64MMFR0_EL1 bit assignments

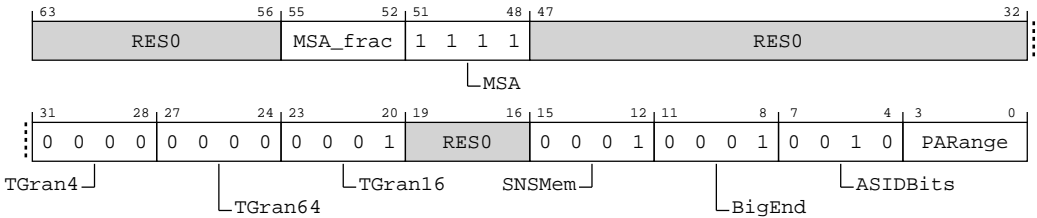


Table A-80: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:52]	MSA_frac	<p>When VMSAm == 1</p> <p>Memory System Architecture fractional field. This holds information on additional Memory System Architectures supported.</p> <p>0010</p> <p>PMSAv8-64 supported in all translation regimes. In addition to PMSAv8-64, stage 1 EL1&0 transaltion regime also supports VMSAv8-64.</p> <p>Otherwise</p> <p>Memory System Architecture fractional field. This holds information on additional Memory System Architectures supported.</p> <p>0001</p> <p>PMSAv8-64 supported in all translation regimes. VMSAv8-64 not supported.</p>	xxxx
[51:48]	MSA	<p>Memory System Architecture ID field. This holds the information on Memory System Architectures supported. Defined values are:</p> <p>0b1111</p> <p>See ID_AA64MMFR0_EL1.MSA_frac for the Memory System Architectures supported.</p>	0b1111
[47:32]	RES0	Reserved	RES0
[31:28]	TGran4	<p>Indicates support for 4KB memory translation granule size. Defined values are:</p> <p>0b0000</p> <p>4KB granule supported.</p>	0b0000
[27:24]	TGran64	<p>Indicates support for 64KB memory translation granule size. Defined values are:</p> <p>0b0000</p> <p>64KB granule supported.</p>	0b0000
[23:20]	TGran16	<p>Indicates support for 16KB memory translation granule size. Defined values are:</p> <p>0b0001</p> <p>16KB granule supported.</p>	0b0001
[19:16]	RES0	Reserved	RES0
[15:12]	SNSMem	<p>Indicates support for a distinction between Secure and Non-secure Memory. Defined values are:</p> <p>0b0001</p> <p>Does support a distinction between Secure and Non-secure Memory.</p>	0b0001
[11:8]	BigEnd	<p>Indicates support for mixed-endian configuration. Defined values are:</p> <p>0b0001</p> <p>Mixed-endian support. The SCTLRL_ELx.EE and AArch64-SCTLR_EL1.E0E bits can be configured.</p>	0b0001
[7:4]	ASIDBits	<p>Number of ASID bits. Defined values are:</p> <p>0b0010</p> <p>16 bits.</p>	0b0010

Bits	Name	Description	Reset
[3:0]	PARange	When PA_W == 48 Physical Address range supported. 0101 48 bits, 256TB. Otherwise Physical Address range supported. 0010 40 bits, 1TB.	xxxx

Access

MRS <Xt>, ID_AA64MMFRO_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b000

Accessibility

MRS <Xt>, ID_AA64MMFRO_EL1

```

if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFRO_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFRO_EL1;

```

A.2.1.36 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

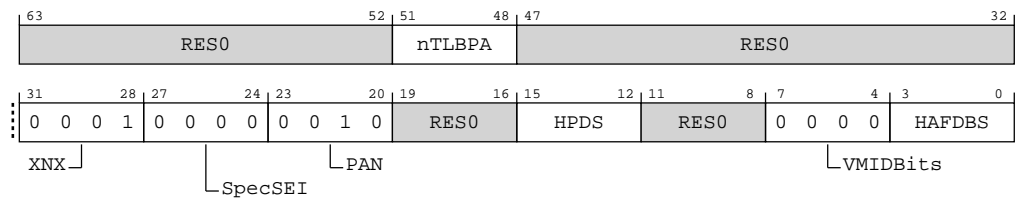
See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	0000	0010	xxxx	xxxx	xxxx	0000	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-36: AARCH64_ID_AA64MMFR1_EL1 bit assignments****Table A-82: ID_AA64MMFR1_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:52]	RES0	Reserved	RES0
[51:48]	nTLBPA	When VMSAm == 1 Indicates support for intermediate caching of translation table walks. 0001 The intermediate caching of translation table walks does not include non-coherent caches of previous valid translation table entries since the last completed TLBI applicable to the PE where either: <ul style="list-style-type: none"> The caching is indexed by the physical address of the location holding the translation table entry. The caching is used for stage 1 translations and is indexed by the intermediate physical address of the location holding the translation table entry. Otherwise RES0	xxxx
[47:32]	RES0	Reserved	RES0
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are: 0b0001 Distinction between EL0 and EL1 execute-never control at stage 2 supported.	0b0001

Bits	Name	Description	Reset
[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, and AArch64-DSPSR_ELO. Defined values are: 0b0010 PAN supported and AT S1E1RP and AT S1E1WP instructions supported.	0b0010
[19:16]	RES0	Reserved	RES0
[15:12]	HPDS	When VMSAm == 1 Hierarchical permission disables bits in translation tables. 0001 Disabling of hierarchical controls supported with the AArch64-TCR_EL1.{HPD1, HPD0} bits. Otherwise Hierarchical permission disables bits in translation tables. 0000 Disabling of hierarchical controls not supported.	xxxx
[11:8]	RES0	Reserved	RES0
[7:4]	VMIDBits	Number of VMID bits. Defined values are: 0b0000 8 bits	0b0000
[3:0]	HAFDBS	When VMSAm == 1 Hardware updates to Access flag and Dirty state in translation tables. 0001 Hardware update of the Access flag is supported. Otherwise Hardware updates to Access flag and Dirty state in translation tables. 0000 Hardware update of the Access flag and dirty state are not supported.	xxxx

Access

MRS <Xt>, ID_AA64MMFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b001

Accessibility

MRS <Xt>, ID_AA64MMFR1_EL1

```

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

```


```
elseif PSTATE.EL == EL1 then
  if HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    X[t, 64] = ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL2 then
  X[t, 64] = ID_AA64MMFR1_EL1;
```

A.2.1.37 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations


Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group


Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	0001	0001	0001	xxxx	xxxx	0000	xxxx	0001	xxxx	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 A-37: AARCH64_ID_AA64MMFR2_EL1 bit assignments

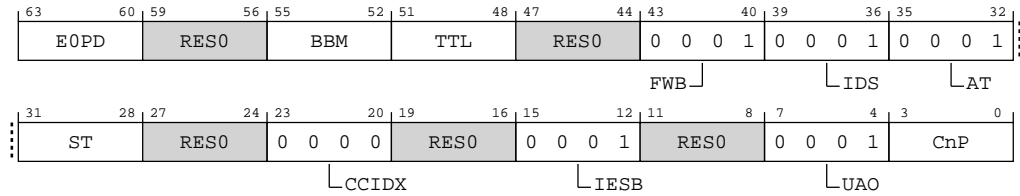


Table A-84: ID_AA64MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	EOPD	When VMSAm == 1 Indicates support for the FEAT_EOPD mechanism. 0001 EOPDx mechanism is implemented. Otherwise Indicates support for the FEAT_EOPD mechanism. 0000 EOPDx mechanism is not implemented.	xxxx
[59:56]	RES0	Reserved	RES0
[55:52]	BBM	When VMSAm == 1 Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. 0001 Level 1 support for changing block size is supported. Otherwise Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. 0000 Level 0 support for changing block size is supported.	xxxx
[51:48]	TTL	When VMSAm == 1 Indicates support for TTL field in address operations. 0001 TLB maintenance instructions by address have bits[47:44] holding the TTL field. Otherwise Indicates support for TTL field in address operations. 0000 TLB maintenance instructions by address have bits[47:44] as RES0.	xxxx
[47:44]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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. The Feature ID space is defined as the System register space in AArch64 with op0==3, op1=={0, 1, 3}, CRn==0, CRm=={0-7}, op2=={0-7}.	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	When VMSAm == 1 Identifies support for small translation tables. 0001 The maximum value of the TCR_ELx.{TOSZ,T1SZ} and VTCR_EL2.TOSZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules. Otherwise Identifies support for small translation tables. 0000 The maximum value of the TCR_ELx.{TOSZ,T1SZ} and VTCR_EL2.TOSZ fields is 39.	xxxx
[27:24]	RES0	Reserved	RES0
[23:20]	CCIDX	Support for the use of revised AArch64-CCSIDR_EL1 register format. Defined values are: 0b0000 32-bit format implemented for all levels of the CCSIDR_EL1.	0b0000
[19:16]	RES0	Reserved	RES0
[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]	RES0	Reserved	RES0
[7:4]	UAO	User Access Override. Defined values are: 0b0001 UAO supported.	0b0001

Bits	Name	Description	Reset
[3:0]	CnP	When VMSAm == 1 Common not Private translations. 0001 Common not Private translations supported. Otherwise Common not Private translations. 0000 Common not Private translations not supported.	xxxx

Access

MRS <Xt>, ID_AA64MMFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0111	0b010

Accessibility

MRS <Xt>, ID_AA64MMFR2_EL1

```

if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR2_EL1;

```

A.2.1.38 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

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-38: AARCH64_ID_AA64MMFR3_EL1 bit assignments

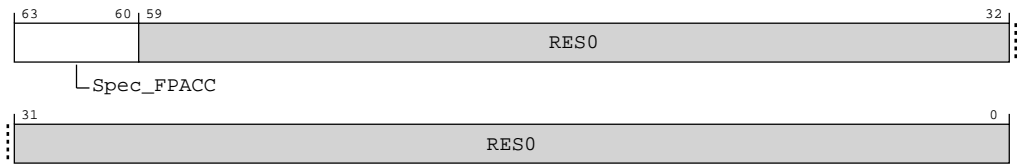


Table A-86: ID_AA64MMFR3_EL1 bit descriptions

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: 0b0001 The speculative use of pointers processed by a PAC Authentication is not materially different in terms of the impact on cached microarchitectural state between passing and failing of the PAC Authentication. All other values are reserved. For the purpose of this definition, cached microarchitecture state is the state of caching agents such as instruction caches, data caches and TLBs which can be altered as a result of speculation caused by a mispredicted execution, but is not restored to the state prior to the speculation when the misprediction is corrected.	xxxx
[59:0]	RES0	Reserved	RES0

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 HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64MMFR3_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64MMFR3_EL1;

```

A.2.1.39 CCSIDR_EL1, Current Cache Size ID Register

Provides information about the architecture of the currently selected cache.

Configurations

The implementation includes one CCSIDR_EL1 for each cache that it can access. AArch64-CSSELR_EL1 selects which Cache Size ID Register is accessible.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

When AArch64-CSSELR_EL1.Level == '000' and AArch64-CSSELR_EL1.InD == '0'

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0001 1010

When AArch64-CSSELR_EL1.Level == '000' and AArch64-CSSELR_EL1.InD == '1'

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0001 1010

When L2_CACHE_SIZE > 0, AArch64-CSSELR_EL1.Level == '001' and AArch64-CSSELR_EL1.InD == '0'

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx0 0000 0001 1010

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-CSSELR_EL1.Level == '000' and AArch64-CSSELR_EL1.InD == '0'

Figure A-39: AARCH64_CCSIDR_EL1 bit assignments

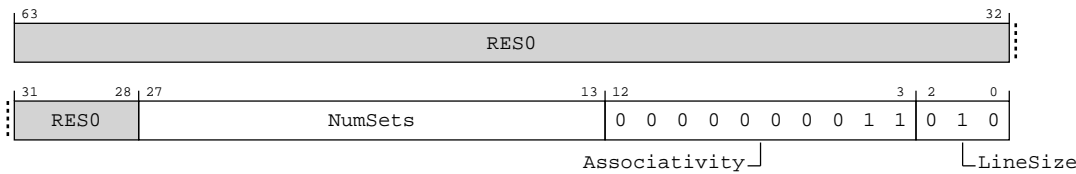


Table A-88: CCSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:28]	RES0	Reserved	RES0
[27:13]	NumSets	(Number of sets in cache) - 1. 0b0000000001111111 64 sets. 0b0000000011111111 128 sets. 0b0000000111111111 256 sets.	15{x}
[12:3]	Associativity	(Associativity of cache) - 1. 0b0000000011 4 cache lines per set.	0b0000000011
[2:0]	LineSize	(Log ₂ (Number of bytes in cache line)) - 4. 0b010 64 bytes per cache line.	0b010

When AArch64-CSSELR_EL1.Level == '000' and AArch64-CSSELR_EL1.InD == '1'

Figure A-40: AARCH64_CCSIDR_EL1 bit assignments

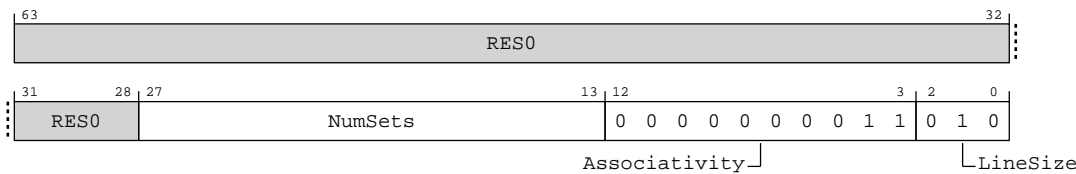


Table A-89: CCSIDR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[27:13]	NumSets	(Number of sets in cache) - 1. 0b000000000111111 64 sets. 0b000000001111111 128 sets. 0b000000011111111 256 sets. 0b000000111111111 512 sets.	15{x}
[12:3]	Associativity	(Associativity of cache) - 1. 0b0000000011 4 cache lines per set.	0b0000000011
[2:0]	LineSize	(Log ₂ (Number of bytes in cache line)) - 4. 0b010 64 bytes per cache line.	0b010

When L2_CACHE_SIZE > 0, AArch64-CSSELR_EL1.Level == '001' and AArch64-CSSELR_EL1.InD == '0'

Figure A-41: AARCH64_CCSDR_EL1 bit assignments

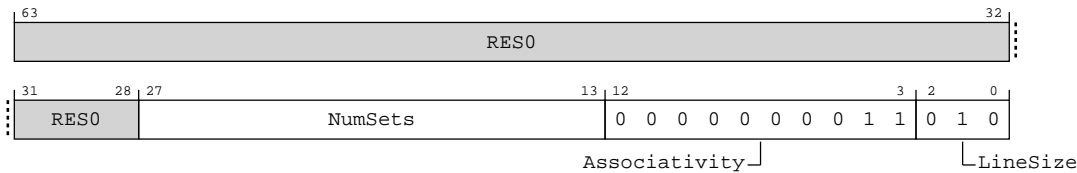


Table A-90: CCSIDR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[27:13]	NumSets	(Number of sets in cache) - 1. 0b0000000101111111 192 sets. 0b0000000111111111 256 sets. 0b0000001011111111 384 sets. 0b0000001111111111 512 sets. 0b0000010111111111 768 sets. 0b0000011111111111 1024 sets. 0b0000101111111111 1536 sets. 0b0000111111111111 2048 sets. 0b0001011111111111 3072 sets. 0b0001111111111111 4096 sets. 0b0010111111111111 6144 sets. 0b0011111111111111 8192 sets.	15{x}
[12:3]	Associativity	(Associativity of cache) - 1. 0b0000000011 4 cache lines per set.	0b0000000011
[2:0]	LineSize	(Log ₂ (Number of bytes in cache line)) - 4. 0b010 64 bytes per cache line.	0b010

Figure A-42: AARCH64_CCSIDR_EL1 bit assignments

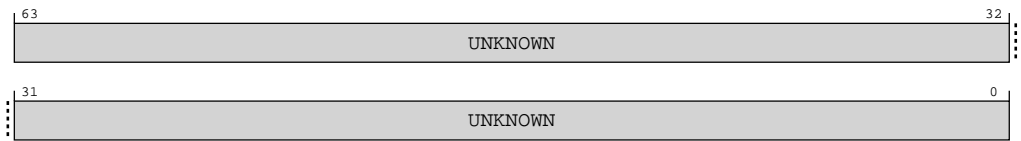


Table A-91: CCSIDR_EL1 bit descriptions

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

Access

If AArch64-CSSELR_EL1.{Level, InD} is programmed to a cache level that is not implemented, then on a read of the CCSIDR_EL1 the behavior is **CONSTRAINED UNPREDICTABLE**, and can be one of the following:

- The CCSIDR_EL1 read returns an **UNKNOWN** value.

MRS <Xt>, CCSIDR_EL1

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

Accessibility

If AArch64-CSSELR_EL1.{Level, InD} is programmed to a cache level that is not implemented, then on a read of the CCSIDR_EL1 the behavior is **CONSTRAINED UNPREDICTABLE**, and can be one of the following:

- The CCSIDR_EL1 read returns an **UNKNOWN** value.

MRS <Xt>, CCSIDR_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CCSIDR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CCSIDR_EL1;
```

A.2.1.40 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

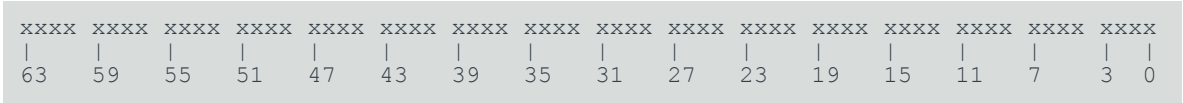
64

Functional group

Identification registers

Access type
See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-43: AARCH64_CLIDR_EL1 bit assignments

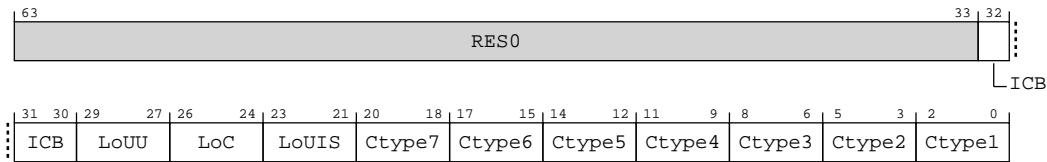


Table A-93: CLIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:33]	RES0	Reserved	RES0
[32:30]	ICB	<div><div>When L2_CACHE_SIZE > 0</div><div>Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.</div><div>010</div><div>L2 cache is the highest Inner Cacheable level.</div><div>Otherwise</div><div>Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.</div><div>001</div><div>L1 cache is the highest Inner Cacheable level.</div></div>	xxx

Bits	Name	Description	Reset
[29:27]	LoUU	<p>Level of Unification Uniprocessor for the cache hierarchy.</p> <p>For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.</p> <p>Note: This field does not describe the requirements for instruction cache invalidation. See AArch64-CTR_ELO.DIC.</p> <p>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.</p> <p>0b000 Level of Unification Uniprocessor is before the L1 cache.</p>	xxx
[26:24]	LoC	<p>When L2_CACHE_SIZE > 0 Level of Coherence for the cache hierarchy.</p> <p>010 L2 cache is the level of coherence.</p> <p>Otherwise Level of Coherence for the cache hierarchy.</p> <p>001 L1 cache is the level of coherence.</p>	xxx
[23:21]	LoUIS	<p>Level of Unification Inner Shareable for the cache hierarchy.</p> <p>For a description of the values of this field, see Terminology for Clean, Invalidate, and Clean and Invalidate instructions.</p> <p>Note: This field does not describe the requirements for instruction cache invalidation. See AArch64-CTR_ELO.DIC.</p> <p>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.</p> <p>0b000 Level of Unification Inner Shareable is before the L1 cache.</p>	xxx
[20:18]	Ctype7	<p>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:</p> <p>0b000 No cache.</p>	xxx
[17:15]	Ctype6	<p>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:</p> <p>0b000 No cache.</p>	xxx

Bits	Name	Description	Reset
[14:12]	Ctype5	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No cache.	xxx
[11:9]	Ctype4	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No cache.	xxx
[8:6]	Ctype3	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b000 No cache. All other values are reserved. If software reads the Cache Type fields from Ctype1 upwards, once it has seen a value of 000, no caches that can be managed using the architected cache maintenance instructions that operate by set/way exist at further-out levels of the hierarchy. So, for example, if Ctype3 is the first Cache Type field with a value of 000, the values of Ctype4 to Ctype7 must be ignored.	xxx
[5:3]	Ctype2	When L2_CACHE_SIZE > 0 Cache Type at level 2. 100 Unified cache at L2. Otherwise Cache Type at level 2. 000 No cache at L2.	xxx
[2:0]	Ctype1	Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are: 0b011 Separate instruction and data caches.	xxx

Access

MRS <Xt>, CLIDR_EL1

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

Accessibility

MRS <Xt>, CLIDR_EL1

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

```
if HCR_EL2.TGE == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CLIDR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = CLIDR_EL1;
```

A.2.1.41 CSSELR_EL1, Cache Size Selection Register

Selects the current Cache Size ID Register, AArch64-CCSIDR_EL1, by specifying the required cache level and the cache type (either instruction or data cache).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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_CSSELR_EL1 bit assignments

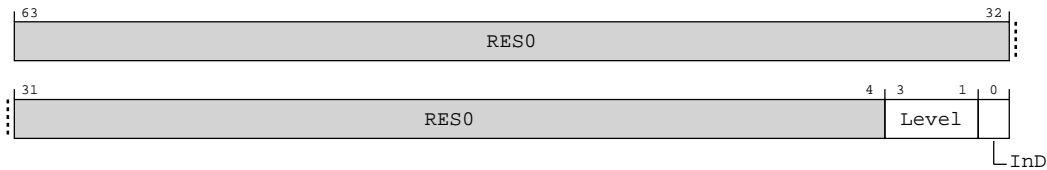


Table A-95: CSSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:1]	Level	<p>Cache level of required cache.</p> <p>0b000 Level 1 cache.</p> <p>0b001 Level 2 cache.</p> <p>0b010 Level 3 cache.</p> <p>0b011 Level 4 cache.</p> <p>0b100 Level 5 cache.</p> <p>0b101 Level 6 cache.</p> <p>0b110 Level 7 cache.</p> <p>All other values are reserved.</p> <p>If CSSELR_EL1.{Level, InD} is programmed to a cache level that is not implemented, then the value for this field on a read of CSSELR_EL1 is UNKNOWN.</p>	xxx
[0]	InD	<p>Instruction not Data bit.</p> <p>0b0 Data or unified cache.</p> <p>0b1 Instruction cache.</p> <p>If CSSELR_EL1.{Level, InD} is programmed to a cache level that is not implemented, then a read of CSSELR_EL1 is CONSTRAINED UNPREDICTABLE, and returns UNKNOWN values for CSSELR_EL1.{Level, InD}.</p>	x

Access

MRS <Xt>, CSSELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b0000	0b0000	0b000

MSR CSSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b010	0b0000	0b0000	0b000

Accessibility

MRS <Xt>, CSSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CSSELR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CSSELR_EL1;
```

MSR CSSELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        CSSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    CSSELR_EL1 = X[t, 64];
```

A.2.1.42 CTR_EL0, Cache Type Register

Provides information about the architecture of the caches.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx01	0100	0100	0100	10xx	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-45: AARCH64_CTR_EL0 bit assignments

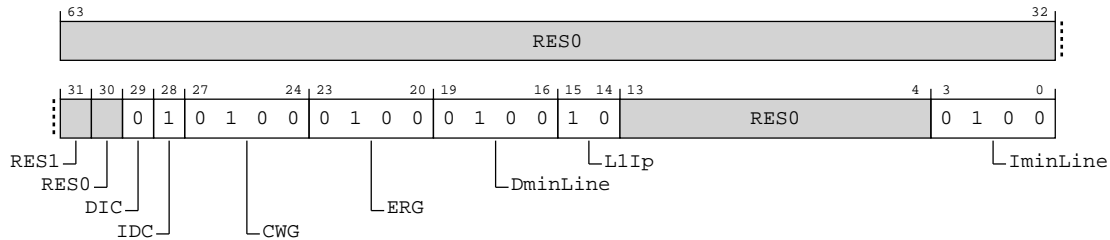


Table A-98: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	RES1	Reserved	RES1
[30]	RES0	Reserved	RES0
[29]	DIC	Instruction cache invalidation requirements for data to instruction coherence. 0b0 Instruction cache invalidation to the Point of Unification is required for data to instruction coherence.	0b0
[28]	IDC	Data cache clean requirements for instruction to data coherence. The meaning of this bit is: 0b1 Data cache clean to the Point of Unification is not required for instruction to data coherence.	0b1
[27:24]	CWG	Cache writeback granule. Log ₂ of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified. 0b0100 64 bytes.	0b0100
[23:20]	ERG	Exclusives reservation granule. Log ₂ of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions. 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: 0b10 Virtual Index, Physical Tag (VIPT).	0b10
[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 SCTL_EL1.UCT == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TID2 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CTR_EL0;
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TID2 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CTR_EL0;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = CTR_EL0;
```

A.2.1.43 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.

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	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-46: AARCH64_DCZID_ELO bit assignments

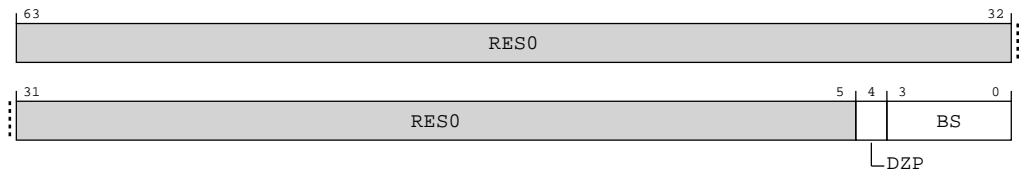


Table A-100: DCZID_ELO bit descriptions

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. 0b0 Instructions are permitted. 0b1 Instructions are prohibited. The value read from this field is governed by the access state and the values of the AArch64-HCR_EL2.TDZ and AArch64-SCTLR_EL1.DZE bits.	x
[3:0]	BS	Log ₂ of the block size in words. The maximum size supported is 2KB, indicated by value 0b1001.	xxxx

Access

MRS <Xt>, DCZID_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, DCZID_ELO

```
if PSTATE.EL == EL0 then
    X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL1 then
    X[t, 64] = DCZID_EL0;
elsif PSTATE.EL == EL2 then
    X[t, 64] = DCZID_EL0;
```

A.2.1.44 VPIDR_EL2, Virtualization Processor ID Register

Holds the value of the Virtualization Processor ID. This is the value returned by EL1 reads of AArch64-MIDR_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

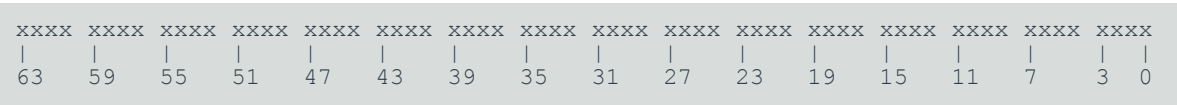
Functional group

Identification registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-47: AARCH64_VPIDR_EL2 bit assignments

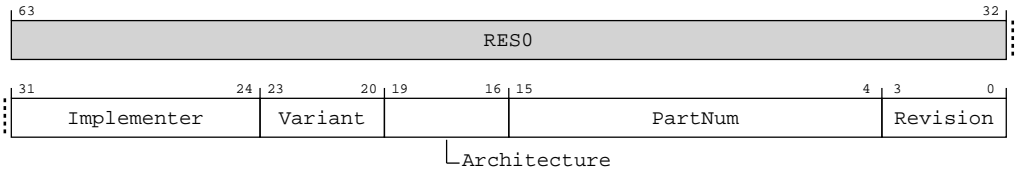


Table A-102: VPIDR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:24]	Implementer	<p>The Implementer code. This field must hold an implementer code that has been assigned by Arm. Assigned codes include the following:</p> <p>0b00000000 Reserved for software use.</p> <p>0b01000001 Arm Limited.</p> <p>0b01000010 Broadcom Corporation.</p> <p>0b01000011 Cavium Inc.</p> <p>0b01000100 Digital Equipment Corporation.</p> <p>0b01000110 Fujitsu Ltd.</p> <p>0b01001001 Infineon Technologies AG.</p> <p>0b01001101 Motorola or Freescale Semiconductor Inc.</p> <p>0b01001110 NVIDIA Corporation.</p> <p>0b01010000 Applied Micro Circuits Corporation.</p> <p>0b01010001 Qualcomm Inc.</p> <p>0b01010110 Marvell International Ltd.</p> <p>0b01101001 Intel Corporation.</p> <p>0b11000000 Ampere Computing.</p> <p>Arm can assign codes that are not published in this manual. All values not assigned by Arm are reserved and must not be used.</p>	8 {x}
[23:20]	Variant	An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.	xxx

Bits	Name	Description	Reset
[19:16]	Architecture	<p>Architecture version. Defined values are:</p> <p>0b0001 Armv4.</p> <p>0b0010 Armv4T.</p> <p>0b0011 Armv5 (obsolete).</p> <p>0b0100 Armv5T.</p> <p>0b0101 Armv5TE.</p> <p>0b0110 Armv5TEJ.</p> <p>0b0111 Armv6.</p> <p>0b1111 Architectural features are individually identified in the ID_* registers.</p> <p>All other values are reserved.</p>	xxxx
[15:4]	PartNum	<p>An IMPLEMENTATION DEFINED primary part number for the device.</p> <p>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.</p>	12{x}
[3:0]	Revision	An IMPLEMENTATION DEFINED revision number for the device.	xxxx

Access

MRS <Xt>, VPIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b000

MSR VPIDR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b000

MRS <Xt>, MIDR_EL1

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

Accessibility

MRS <Xt>, VPIDR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = VPIDR_EL2;
```

MSR VPIDR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    VPIDR_EL2 = X[t, 64];
```

MRS <Xt>, MIDR_EL1

```
if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    X[t, 64] = VPIDR_EL2;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MIDR_EL1;
```

A.2.1.45 VMPIDR_EL2, Virtualization Multiprocessor ID Register

Holds the value of the Virtualization Multiprocessor ID. This is the value returned by EL1 reads of AArch64-MPIDR_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

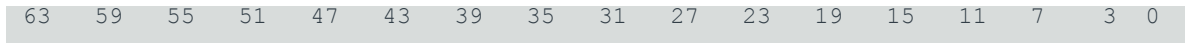
Identification registers

Access type

See bit descriptions

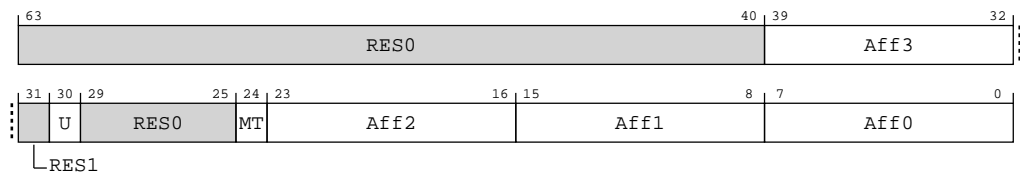
Reset value

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

**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-48: AARCH64_VMPIDR_EL2 bit assignments**Table A-106: VMPIDR_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39:32]	Aff3	Affinity level 3. See the description of VMPIDR_EL2.Aff0 for more information.	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. 0b1 Processor is part of a uniprocessor system.	x
[29:25]	RES0	Reserved	RES0
[24]	MT	Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of VMPIDR_EL2.Aff0 for more information about affinity levels. 0b0 Performance of PEs at the lowest affinity level is largely independent. 0b1 Performance of PEs at the lowest affinity level is very interdependent.	x
[23:16]	Aff2	Affinity level 2. See the description of VMPIDR_EL2.Aff0 for more information.	8 {x}
[15:8]	Aff1	Affinity level 1. See the description of VMPIDR_EL2.Aff0 for more information.	8 {x}
[7:0]	Aff0	Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the AArch64-MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole.	8 {x}

Access

MRS <Xt>, VMPIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b101

MSR VMPIDR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0000	0b0000	0b101

MRS <Xt>, MPIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b101

Accessibility

MRS <Xt>, VMPIDR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VMPIDR_EL2;

```

MSR VMPIDR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    VMPIDR_EL2 = X[t, 64];

```

MRS <Xt>, MPIDR_EL1

```

if PSTATE.EL == EL0 then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    X[t, 64] = VMPIDR_EL2;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MPIDR_EL1;

```

A.2.2 AArch64 Generic System control register description

This section includes the register descriptions for all Generic System control registers in the Cortex®-R82 processor.

A.2.2.1 MPUIR_EL1, MPU Type Register (EL1)

Identifies the number of regions supported by the EL1 MPU.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-49: AARCH64_MPUIR_EL1 bit assignments

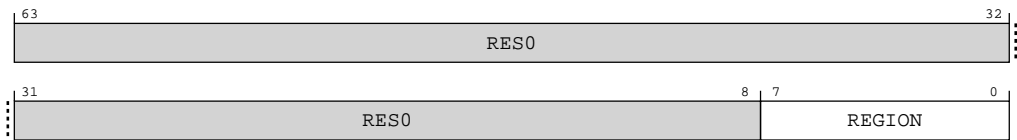


Table A-110: MPUIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	REGION	The number of EL1 MPU regions supported.	8 {x}

Access

MRS <Xt>, MPUIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0000	0b100

Accessibility

MRS <Xt>, MPUIR_EL1

```
if PSTATE.EL == EL0 then
  if HCR_EL2.TGE == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
  if HCR_EL2.TID1 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  elseif VTCR_EL2.MSA == '1' then
    UNDEFINED;
  else
    X[t, 64] = MPUIR_EL1;
elseif PSTATE.EL == EL2 then
  X[t, 64] = MPUIR_EL1;
```

A.2.2.2 SCTRL_EL1, System Control Register (EL1)

Provides top level control of the system, including its memory system, at EL1 and EL0.

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-50: AARCH64_SCTLR_EL1 bit assignments

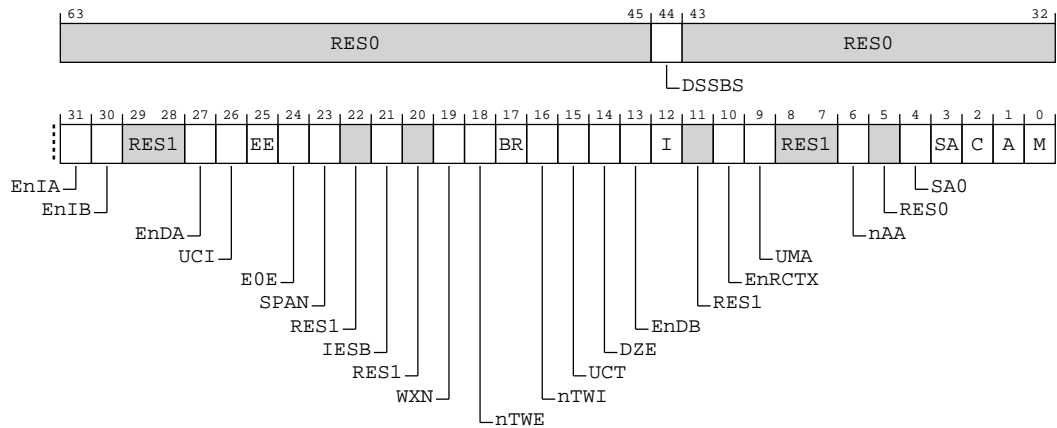


Table A-112: SCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:45]	RES0	Reserved	RES0
[44]	DSSBS	Default PSTATE.SSBS value on Exception Entry. 0b0 PSTATE.SSBS is set to 0 on an exception to EL1. 0b1 PSTATE.SSBS is set to 1 on an exception to EL1.	x
[43:32]	RES0	Reserved	RES0
[31]	EnIA	Controls enabling of pointer authentication of instruction addresses, using the APIAKey_EL1 key, in the EL1&O translation regime. 0b0 Pointer authentication of instruction addresses, using the APIAKey_EL1 key, is not enabled. 0b1 Pointer authentication of instruction addresses, using the APIAKey_EL1 key, is enabled. Note: This field controls the behavior of the AddPACIA and AuthIA pseudocode functions. Specifically, when the field is 1, AddPACIA returns a copy of a pointer to which a pointer authentication code has been added, and AuthIA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP .	x

Bits	Name	Description	Reset
[30]	EnIB	Controls enabling of pointer authentication of instruction addresses, using the APIBKey_EL1 key, in the EL1&O translation regime. 0b0 Pointer authentication of instruction addresses, using the APIBKey_EL1 key, is not enabled. 0b1 Pointer authentication of instruction addresses, using the APIBKey_EL1 key, is enabled. Note: This field controls the behavior of the AddPACIB and AuthIB pseudocode functions. Specifically, when the field is 1, AddPACIB returns a copy of a pointer to which a pointer authentication code has been added, and AuthIB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP .	x
[29:28]	RES1	Reserved	RES1
[27]	EnDA	Controls enabling of pointer authentication of instruction addresses, using the APDAKey_EL1 key, in the EL1&O translation regime. 0b0 Pointer authentication of data addresses, using the APDAKey_EL1 key, is not enabled. 0b1 Pointer authentication of data addresses, using the APDAKey_EL1 key, is enabled. Note: This field controls the behavior of the AddPACDA and AuthDA pseudocode functions. Specifically, when the field is 1, AddPACDA returns a copy of a pointer to which a pointer authentication code has been added, and AuthDA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP .	x
[26]	UCI	Traps ELO execution of cache maintenance instructions, to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18. This applies to DC CVAU, DC CIVAC, DC CVAC, DC CVAP, and IC IVAU. If FEAT_DPB2 is implemented, this trap also applies to DC CVADP. 0b0 Execution of the specified instructions at ELO using AArch64 is trapped. 0b1 This control does not cause any instructions to be trapped.	x
[25]	EE	Endianness of data accesses at EL1, and stage 1 translation table walks in the EL1&O translation regime. 0b0 Explicit data accesses at EL1, and stage 1 translation table walks in the EL1&O translation regime are little-endian. 0b1 Explicit data accesses at EL1, and stage 1 translation table walks in the EL1&O translation regime are big-endian. The EE bit is permitted to be cached in a TLB.	x

Bits	Name	Description	Reset
[24]	EOE	<p>Endianness of data accesses at EL0.</p> <p>0b0 Explicit data accesses at EL0 are little-endian.</p> <p>0b1 Explicit data accesses at EL0 are big-endian.</p> <p>This bit has no effect on the endianness of LDTR, LDTRH, LDTRSH, LDTRSW, STTR, and STTRH instructions executed at EL1.</p>	x
[23]	SPAN	<p>Set Privileged Access Never, on taking an exception to EL1.</p> <p>0b0 PSTATE.PAN is set to 1 on taking an exception to EL1.</p> <p>0b1 The value of PSTATE.PAN is left unchanged on taking an exception to EL1.</p>	x
[22]	RES1	Reserved	RES1
[21]	IESB	<p>Implicit Error Synchronization event enable. Possible values are:</p> <p>0b0 Disabled.</p> <p>0b1 An implicit error synchronization event is added:</p> <ul style="list-style-type: none"> At each exception taken to EL1. Before the operational pseudocode of each ERET instruction executed at EL1. <p>When the PE is in Debug state, this field has no effect.</p>	x
[20]	RES1	Reserved	RES1
[19]	WXN	<p>Write permission implies XN (Execute-never). For the EL1&O translation regime, this bit can force all memory regions that are writable to be treated as XN.</p> <p>0b0 This control has no effect on memory access permissions.</p> <p>0b1 Any region that is writable in the EL1&O translation regime is forced to XN for accesses from software executing at EL1 or EL0.</p> <p>This bit applies only when SCTLR_EL1.M bit is set.</p> <p>The WXN bit is permitted to be cached in a TLB.</p>	x

Bits	Name	Description	Reset
[18]	nTWE	<p>Traps ELO execution of WFE instructions to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, from both Execution states, reported using an ESR_ELx.EC value of 0x01.</p> <p>0b0</p> <p>Any attempt to execute a WFE instruction at ELO is trapped, if the instruction would otherwise have caused the PE to enter a low-power state.</p> <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p> <p>Note:</p> <p>Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.</p>	x
[17]	BR	<p>When AArch64-VTCR_EL2.MSA == '0'</p> <p>Background region enable for EL1 MPU memory regions.</p> <p>0b0</p> <p>Background region disabled for stage 1 EL1&O translation regime.</p> <p>0b1</p> <p>Background region enabled for stage 1 EL1&O translation regime.</p> <p>If the value of AArch64-HCR_EL2.{DC, TGE} is not {0, 0} then PE behaves as if the value of the AArch64-SCTLR_EL1.BR field is 0 for all purposes other than returning the value of a direct read of the field.</p> <p>If EL1 MPU is enabled, then ELO access that does not match an EL1 MPU region always results in a Translation fault.</p> <p>Otherwise</p> <p>RES0</p>	'0'
[16]	nTWI	<p>Traps ELO execution of WFI instructions to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, from both Execution states, reported using an ESR_ELx.EC value of 0x01.</p> <p>0b0</p> <p>Any attempt to execute a WFI instruction at ELO is trapped, if the instruction would otherwise have caused the PE to enter a low-power state.</p> <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p> <p>Note:</p> <p>Since a WFE or WFI can complete at any time, even without a Wakeup event, the traps on WFE or WFI are not guaranteed to be taken, even if the WFE or WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.</p>	x

Bits	Name	Description	Reset
[15]	UCT	<p>Traps ELO accesses to the AArch64-CTR_ELO to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.</p> <p>0b0</p> <p>Accesses to the AArch64-CTR_ELO from EL0 using AArch64 are trapped.</p> <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p>	x
[14]	DZE	<p>Traps ELO execution of DC ZVA instructions to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.</p> <p>0b0</p> <p>Any attempt to execute an instruction that this trap applies to at EL0 using AArch64 is trapped.</p> <p>Reading AArch64-DCZID_ELO.DZP from EL0 returns 1, indicating that the instructions this trap applies to are not supported.</p> <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p>	x
[13]	EnDB	<p>Controls enabling of pointer authentication of instruction addresses, using the APDBKey_EL1 key, in the EL1&0 translation regime.</p> <p>0b0</p> <p>Pointer authentication of data addresses, using the APDBKey_EL1 key, is not enabled.</p> <p>0b1</p> <p>Pointer authentication of data addresses, using the APDBKey_EL1 key, is enabled.</p> <p>Note:</p> <p>This field controls the behavior of the AddPACDB and AuthDB pseudocode functions. Specifically, when the field is 1, AddPACDB returns a copy of a pointer to which a pointer authentication code has been added, and AuthDB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP.</p>	x

Bits	Name	Description	Reset
[12]	I	<p>Stage 1 instruction access Cacheability control, for accesses at EL0 and EL1:</p> <p>0b0</p> <p>All instruction access to Stage 1 Normal memory from EL0 and EL1 are Stage 1 Non-cacheable.</p> <p>If stage 1 EL1&0 translation is in VMSAv8-64 context and the value of SCTLR_EL1.M is 0, then instruction accesses from stage 1 are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.</p> <p>If stage 1 EL1&0 translation is in PMSAv8-64 context and the value of SCTLR_EL1.{BR, M} = {0, 0}, then instruction accesses from stage 1 are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.</p> <p>0b1</p> <p>This control has no effect on the Stage 1 Cacheability of instruction access to Stage 1 Normal memory from EL0 and EL1.</p> <p>If stage 1 EL1&0 translation is in VMSAv8-64 context and the value of SCTLR_EL1.M is 0, then instruction accesses from stage 1 are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.</p> <p>If stage 1 EL1&0 translation is in PMSAv8-64 context, and the value of SCTLR_EL1.{BR, M} = {0, 0}, then instruction accesses from stage 1 are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.</p> <p>When the value of the AArch64-HCR_EL2.DC bit is 1, then instruction access to Normal memory from EL0 and EL1 are Cacheable regardless of the value of the SCTLR_EL1.I bit.</p>	x
[11]	RES1	Reserved	RES1
[10]	EnRCTX	<p>Enable EL0 access to the following System instructions:</p> <ul style="list-style-type: none"> CFP RCTX, DVP RCTX and CPP RCTX instructions. <p>0b0</p> <p>EL0 access to these instructions is disabled, and these instructions are trapped to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1.</p> <p>0b1</p> <p>EL0 access to these instructions is enabled.</p>	x
[9]	UMA	<p>User Mask Access. Traps EL0 execution of MSR and MRS instructions that access the PSTATE.{D, A, I, F} masks to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, from AArch64 state only, reported using an ESR_ELx.EC value of 0x18.</p> <p>0b0</p> <p>Any attempt at EL0 using AArch64 to execute an MRS, MSR(register), or MSR(immediate) instruction that accesses the AArch64-DAIF is trapped.</p> <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p>	x
[8:7]	RES1	Reserved	RES1

Bits	Name	Description	Reset
[6]	nAA	<p>Non-aligned access. This bit controls generation of Alignment faults at EL1 and EL0 under certain conditions.</p> <p>The following instructions generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for access:</p> <ul style="list-style-type: none"> LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH. STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH. <p>0b0</p> <p>Unaligned accesses by the specified instructions generate an Alignment fault.</p> <p>0b1</p> <p>This control does not generate Alignment faults.</p>	x
[5]	RES0	Reserved	RES0
[4]	SA0	SP Alignment check enable for EL0. When set to 1, if a load or store instruction executed at EL0 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then an SP alignment fault exception is generated. For more information, see <i>SP alignment checking</i> in the Arm® Architecture Reference Manual for A-profile architecture .	x
[3]	SA	SP Alignment check enable. When set to 1, if a load or store instruction executed at EL1 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then an SP alignment fault exception is generated. For more information, see <i>SP alignment checking</i> in the Arm® Architecture Reference Manual for A-profile architecture .	x
[2]	C	<p>Stage 1 Cacheability control, for data accesses.</p> <p>0b0</p> <p>All data access to Stage 1 Normal memory from EL0 and EL1, and all Normal memory accesses from unified cache to the EL1&0 Stage 1 translation tables, are treated as Stage 1 Non-cacheable.</p> <p>0b1</p> <p>This control has no effect on the Stage 1 Cacheability of:</p> <ul style="list-style-type: none"> Data access to Normal memory from EL0 and EL1. Normal memory accesses to the EL1&0 Stage 1 translation tables. <p>When the Effective value of the HCR_EL2.DC bit in the current Security state is 1, the PE ignores SCTLR_EL1.C. This means that EL0 and EL1 data accesses to Normal memory are Cacheable.</p>	x
[1]	A	<p>Alignment check enable. This is the enable bit for Alignment fault checking at EL1 and EL0.</p> <p>0b0</p> <p>Alignment fault checking is disabled when executing at EL1 or EL0.</p> <p>Alignment checks on some instructions are not disabled by this control. For more information, see <i>Alignment of data accesses</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>0b1</p> <p>Alignment fault checking is enabled when executing at EL1 or EL0.</p> <p>All instructions that load or store one or more registers have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, which is taken as a Data Abort exception.</p>	x

Bits	Name	Description	Reset
[0]	M	<p>MMU or MPU enable for EL1&O stage 1 address translation.</p> <p>This is the enable bit for:</p> <ul style="list-style-type: none"> MPU, if stage 1 EL1&O translation is in PMSAv8-64 context. MMU, if stage 1 EL1&O translation is in VMSAv8-64 context. <p>0b0</p> <p>EL1 MPU(PMSAv8-64) or MMU(VMSAv8-64) disabled</p> <p>See the SCTLR_EL1.I field for the behavior of instruction accesses to Normal memory.</p> <p>0b1</p> <p>EL1 MPU(PMSAv8-64) or MMU(VMSAv8-64) enabled</p> <p>If the Effective value of HCR_EL2.{DC, TGE} in the current Security state is not {0, 0} then the PE behaves as if the value of the SCTLR_EL1.M field is 0 for all purposes other than returning the value of a direct read of the field.</p>	x

Access

MRS <Xt>, SCTLR_EL1

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

MSR SCTLR_EL1, <Xt>

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

Accessibility

MRS <Xt>, SCTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = SCTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SCTLR_EL1;

```

MSR SCTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then

```

```
SCTLR_EL1 = X[t, 64];
```

A.2.2.3 ACTLR_EL1, Auxiliary Control Register (EL1)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

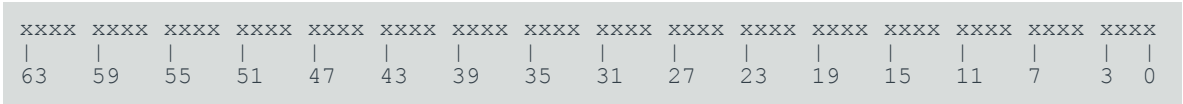
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-51: AARCH64_ACTLR_EL1 bit assignments

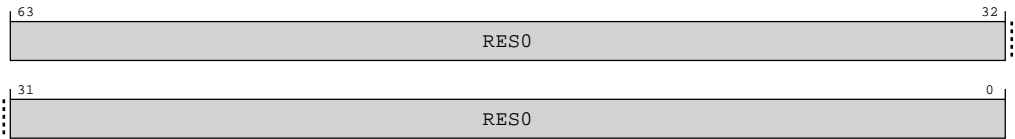


Table A-115: ACTLR_EL1 bit descriptions

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

MSR ACTLR_EL1, <Xt>

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

A.2.2.4 CPACR_EL1, Architectural Feature Access Control Register

Controls access to trace,
and Advanced SIMD and floating-point functionality.

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-52: AARCH64_CPACR_EL1 bit assignments

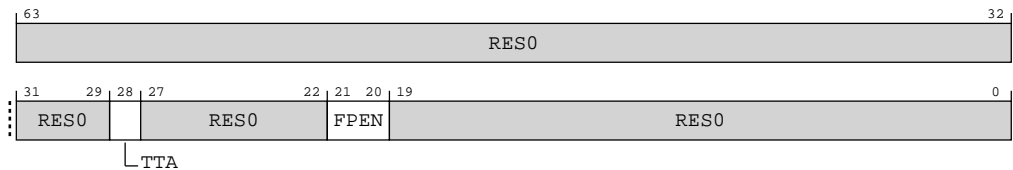


Table A-118: CPACR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:29]	RES0	Reserved	RES0
[28]	TTA	<p>Traps EL0 and EL1 System register accesses to all implemented trace registers from both Execution states to EL1, or to EL2 when it is implemented and enabled in the current Security state and AArch64-HCR_EL2.TGE is 1, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to trace registers are trapped, reported using ESR_ELx.EC value 0x18. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>This control causes EL0 and EL1 System register accesses to all implemented trace registers to be trapped.</p> <ul style="list-style-type: none"> The ETMv4 architecture does not permit EL0 to access the trace registers. If the trace unit implements FEAT_ETMv4, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of AArch64-CPACR_EL1.TTA is 1. The Armv8-A architecture does not provide traps on trace register accesses through the optional memory-mapped interface. <p>System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.</p> <p>If System register access to the trace functionality is not implemented, this bit is RES0.</p>	x
[27:22]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[21:20]	FPEN	<p>Traps execution at EL1 and EL0 of instructions that access the Advanced SIMD and floating-point registers from both Execution states to EL1, reported using ESR_ELx.EC value 0x07, or to EL2 reported using ESR_ELx.EC value 0x00 when EL2 is implemented and enabled in the current Security state and AArch64-HCR_EL2.TGE is 1, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to AArch64-FPCR, AArch64-FPSR, any of the SIMD and floating-point registers V0-V31, including their views as D0-D31 registers or S0-31 registers. <p>0b00 This control causes execution of these instructions at EL1 and EL0 to be trapped.</p> <p>0b01 This control causes execution of these instructions at EL0 to be trapped, but does not cause execution of any instructions at EL1 to be trapped.</p> <p>0b10 This control causes execution of these instructions at EL1 and EL0 to be trapped.</p> <p>0b11 This control does not cause execution of any instructions to be trapped.</p>	xx
[19:0]	RES0	Reserved	RES0

Access

MRS <Xt>, CPACR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b010

MSR CPACR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b010

Accessibility

MRS <Xt>, CPACR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPTR_EL2.TCPAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CPACR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CPACR_EL1;

```

MSR CPACR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPTR_EL2.TCPAC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else

```

```
CPACR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    CPACR_EL1 = X[t, 64];
```

A.2.2.5 TTBR0_EL1, Translation Table Base Register 0 (EL1)

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the lower VA range in the EL1&O translation regime, and other information for this translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-53: AARCH64_TTBR0_EL1 bit assignments

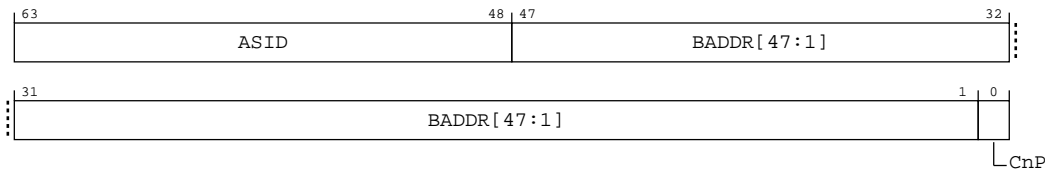


Table A-121: TTBR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	<p>When AArch64-VTCR_EL2.MSA == '1' An ASID for the translation table base address. The AArch64-TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.</p> <p>When AArch64-VTCR_EL2.MSA == '0' An ASID for addresses defined by the current EL1 MPU configuration.</p> <p>Otherwise RES0</p>	xxxx
[47:1]	BADDR[47:1]	<p>When AArch64-VTCR_EL2.MSA == '1' Translation table base address:</p> <ul style="list-style-type: none"> Bits A[47:x] of the stage 1 translation table base address bits are in register bits[47:x]. - Bits A[(x-1):0] of the stage 1 translation table base address are zero. <p>Address bit x is the minimum address bit required to align the translation table to the size of the table. The smallest permitted value of x is 6. The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of AArch64-TCR_EL1.TOSZ, the translation stage, and the translation granule size.</p> <p>Note: A translation table is required to be aligned to the size of the table. If a table contains fewer than eight entries, it must be aligned on a 64 byte address boundary.</p> <p>Otherwise RES0</p>	xxxx

Bits	Name	Description	Reset
[0]	CnP	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Common not Private. This bit indicates whether each entry that is pointed to by TTBR0_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0_EL1.CnP is 1.</p> <p>0b0</p> <p>The translation table entries pointed to by TTBR0_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR0_EL1 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> • The value of TTBR0_EL1.CnP on those other PEs. • The value of the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID. <p>0b1</p> <p>The translation table entries pointed to by TTBR0_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL1.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> • The translation table entries are pointed to by TTBR0_EL1. • The translation tables relate to the same translation regime. • The ASID is the same as the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID. <p>This bit is permitted to be cached in a TLB.</p> <p>When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.</p> <p>Note:</p> <p>If the value of the TTBR0_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>When AArch64-VTCR_EL2.MSA == '0'</p> <p>RES0</p> <p>Otherwise</p> <p>RES0</p>	x

Access

MRS <Xt>, TTBR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

MSR TTBR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b000

Accessibility

MRS <Xt>, TTBR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TTBR0_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TTBR0_EL1<63:0>;
```

MSR TTBR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TTBR0_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL2 then
    TTBR0_EL1<63:0> = X[t, 64];
```

A.2.2.6 TTBR1_EL1, Translation Table Base Register 1 (EL1)

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the higher VA range in the EL1&O stage 1 translation regime, and other information for this translation regime.

Configurations

In a PMSAv8-64 only implementation, this register is UNDEFINED.

This register is present only when ArchHasVMSAExtension(). Otherwise, direct accesses to TTBR1_EL1 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	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-54: AARCH64_TTBR1_EL1 bit assignments

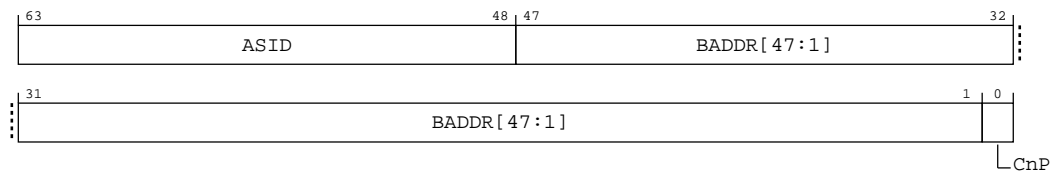


Table A-124: TTBR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>An ASID for the translation table base address. The AArch64-TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.</p> <p>If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.</p> <p>Otherwise</p> <p>RES0</p>	xxxx
[47:1]	BADDR[47:1]	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Translation table base address:</p> <ul style="list-style-type: none"> Bits A[47:x] of the stage 1 translation table base address bits are in register bits[47:x]. - Bits A[(x-1):0] of the stage 1 translation table base address are zero. <p>Address bit x is the minimum address bit required to align the translation table to the size of the table. The smallest permitted value of x is 6. The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of AArch64-TCR_EL1.T1SZ, the translation stage, and the translation granule size.</p> <p>Note:</p> <p>A translation table is required to be aligned to the size of the table. If a table contains fewer than eight entries, it must be aligned on a 64 byte address boundary.</p> <p>Otherwise</p> <p>RES0</p>	xxxx

Bits	Name	Description	Reset
[0]	CnP	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Common not Private. This bit indicates whether each entry that is pointed to by TBR1_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1_EL1.CnP is 1.</p> <p>0b0</p> <p>The translation table entries pointed to by TTBR1_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR1_EL1 for other PEs in the Inner Shareable domain. This is not affected by:</p> <ul style="list-style-type: none"> • The value of TTBR1_EL1.CnP on those other PEs. • The value of the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID. <p>0b1</p> <p>The translation table entries pointed to by TTBR1_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1_EL1.CnP is 1 and all of the following apply:</p> <ul style="list-style-type: none"> • The translation table entries are pointed to by TTBR1_EL1. • The translation tables relate to the same translation regime. • The ASID is the same as the current ASID. • If EL2 is implemented and enabled in the current Security state, the value of the current VMID. <p>This bit is permitted to be cached in a TLB.</p> <p>When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.</p> <p>Note:</p> <p>If the value of the TTBR1_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>Otherwise</p> <p>RES0</p>	xxxx

Access

MRS <Xt>, TTBR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

MSR TTBR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b001

Accessibility

MRS <Xt>, TTBR1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ID_AA64MMFR0_EL1.MSA_frac == '0001' then
        UNDEFINED;
    elsif HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif VTCR_EL2.MSA == '0' then
        UNDEFINED;
    else
        X[t, 64] = TTBR1_EL1<63:0>;
elsif PSTATE.EL == EL2 then
    if ID_AA64MMFR0_EL1.MSA_frac == '0001' then
        UNDEFINED;
    else
        X[t, 64] = TTBR1_EL1<63:0>;

```

MSR TTBR1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ID_AA64MMFR0_EL1.MSA_frac == '0001' then
        UNDEFINED;
    elsif HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif VTCR_EL2.MSA == '0' then
        UNDEFINED;
    else
        TTBR1_EL1<63:0> = X[t, 64];
elsif PSTATE.EL == EL2 then
    if ID_AA64MMFR0_EL1.MSA_frac == '0001' then
        UNDEFINED;
    else
        TTBR1_EL1<63:0> = X[t, 64];

```

A.2.2.7 TCR_EL1, Translation Control Register (EL1)

The control register for stage 1 of the EL1&0 translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Any of the bits in TCR_EL1, other than the A1 bit and the EPDx bits when they have the value 1, are permitted to be cached in a TLB.

Figure A-55: AARCH64_TCR_EL1 bit assignments

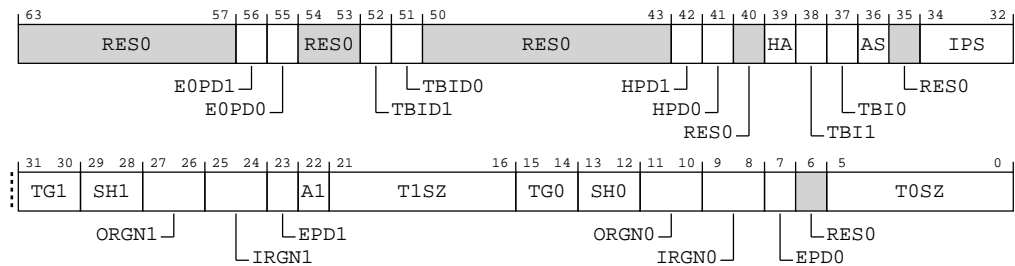


Table A-127: TCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:57]	RES0	Reserved	RES0
[56]	EOPD1	When AArch64-VTCR_EL2.MSA == '1' Faulting control for Unprivileged access to any address translated by AArch64-TTBR1_EL1. 0b0 Unprivileged access to any address translated by AArch64-TTBR1_EL1 will not generate a fault by this mechanism. 0b1 Unprivileged access to any address translated by AArch64-TTBR1_EL1 will generate a level 0 Translation fault. Level 0 Translation faults generated as a result of this field are not counted as TLB misses for performance monitoring. The fault should take the same time to generate, whether the address is present in the TLB or not, to mitigate attacks that use fault timing. Otherwise RES0	xxxx

Bits	Name	Description	Reset
[55]	EOPD0	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Faulting control for Unprivileged access to any address translated by AArch64-TTBR0_EL1.</p> <p>0b0</p> <p>Unprivileged access to any address translated by AArch64-TTBR0_EL1 will not generate a fault by this mechanism.</p> <p>0b1</p> <p>Unprivileged access to any address translated by AArch64-TTBR0_EL1 will generate a level 0 Translation fault.</p> <p>Level 0 Translation faults generated as a result of this field are not counted as TLB misses for performance monitoring. The fault should take the same time to generate, whether the address is present in the TLB or not, to mitigate attacks that use fault timing.</p> <p>Otherwise</p> <p>RES0</p>	xxxx
[54:53]	RES0	Reserved	RES0
[52]	TBID1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Controls the use of the top byte of instruction addresses for address matching.</p> <p>For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses.</p> <p>For more information, see <i>Address tagging</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>0b0</p> <p>TCR_EL1.TBI1 applies to Instruction and Data accesses.</p> <p>0b1</p> <p>TCR_EL1.TBI1 applies to Data accesses only.</p> <p>This affects addresses where the address would be translated by tables pointed to by AArch64-TTBR1_EL1.</p> <p>Otherwise</p> <p>RES0</p>	xxxx

Bits	Name	Description	Reset
[51]	TBID0	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Controls the use of the top byte of instruction addresses for address matching. For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses. For more information, see <i>Address tagging</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>0b0 TCR_EL1.TBID0 applies to Instruction and Data accesses.</p> <p>0b1 TCR_EL1.TBID0 applies to Data accesses only.</p> <p>This affects addresses where the address would be translated by tables pointed to by AArch64-TTBR0_EL1.</p> <p>When AArch64-VTCR_EL2.MSA == '0'</p> <p>Controls the use of the top byte of instruction addresses for address matching. For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses. For more information, see <i>Address tagging</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>0b0 TCR_EL1.TBID0 applies to Instruction and Data accesses.</p> <p>0b1 TCR_EL1.TBID0 applies to Data accesses only.</p> <p>This affects addresses where the address would be translated by EL1 MPU.</p> <p>Otherwise RES0</p>	xxxx
[50:43]	RES0	Reserved	RES0
[42]	HPD1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Hierarchical Permission Disables. This affects the hierarchical control bits, APTTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by AArch64-TTBR1_EL1.</p> <p>0b0 Hierarchical permissions are enabled.</p> <p>0b1 Hierarchical permissions are disabled.</p> <p>When disabled, the permissions are treated as if the bits are zero.</p> <p>Otherwise RES0</p>	xxxx

Bits	Name	Description	Reset
[41]	HPD0	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by AArch64-TTBR0_EL1.</p> <p>0b0</p> <p>Hierarchical permissions are enabled.</p> <p>0b1</p> <p>Hierarchical permissions are disabled.</p> <p>When disabled, the permissions are treated as if the bits are zero.</p> <p>Otherwise</p> <p>RES0</p>	xxxx
[40]	RES0	Reserved	RES0
[39]	HA	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Hardware Access flag update in stage 1 translations from ELO and EL1.</p> <p>0b0</p> <p>Stage 1 Access flag update disabled.</p> <p>0b1</p> <p>Stage 1 Access flag update enabled.</p> <p>When AArch64-VTCR_EL2.MSA == '0'</p> <p>RES0</p> <p>Otherwise</p> <p>RES0</p>	x

Bits	Name	Description	Reset
[38]	TBI1	<p>When AArch64-VTCR_EL2.MSA == '0' RES0</p> <p>When AArch64-VTCR_EL2.MSA == '1' Top Byte ignored. Indicates whether the top byte of an address is used for address match for the AArch64-TTBR1_EL1 region, or ignored and used for tagged addresses.</p> <p>0b0 Top Byte used in the address calculation.</p> <p>0b1 Top Byte ignored in the address calculation.</p> <p>This affects addresses generated in EL0 and EL1 using AArch64 where the address would be translated by tables pointed to by AArch64-TTBR1_EL1. It has an effect whether the EL1&O translation regime is enabled or not.</p> <p>If FEAT_PAuth is implemented and TCR_EL1.TBID1 is 1, then this field only applies to Data accesses.</p> <p>Otherwise, if the value of TBI1 is 1 and bit [55] of the target address to be stored to the PC is 1, then bits[63:56] of that target address are also set to 1 before the address is stored in the PC, in the following cases:</p> <ul style="list-style-type: none"> • A branch or procedure return within EL0 or EL1. • An exception taken to EL1. • An exception return to EL0 or EL1. <p>Otherwise RES0</p>	x

Bits	Name	Description	Reset
[37]	TBIO	<p>When AArch64-VTCR_EL2.MSA == '0'</p> <p>Top Byte ignored. Indicates whether the top byte of an address is used for an address match for the EL1 MPU regions, or ignored and used for tagged addresses.</p> <p>0b0</p> <p>Top Byte used in the address calculation.</p> <p>0b1</p> <p>Top Byte ignored in the address calculation.</p> <p>This affects addresses generated in EL0 and EL1, where the address would be translated by the EL1 MPU.</p> <p>If FEAT_PAuth is implemented and TCR_EL1.TBID0 is 1, then this field only applies to Data accesses.</p> <p>Otherwise, if the value of TBIO is 1 and bit [55] of the target address to be stored to the PC is 0, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:</p> <ul style="list-style-type: none"> • A branch or procedure return within EL0 or EL1. • An exception taken to EL1. • An exception return to EL0 or EL1. <p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Top Byte ignored. Indicates whether the top byte of an address is used for address match for the AArch64-TTBRO_EL1 region, or ignored and used for tagged addresses.</p> <p>0b0</p> <p>Top Byte used in the address calculation.</p> <p>0b1</p> <p>Top Byte ignored in the address calculation.</p> <p>This affects addresses generated in EL0 and EL1 using AArch64 where the address would be translated by tables pointed to by AArch64-TTBRO_EL1. It has an effect whether the EL1&O translation regime is enabled or not.</p> <p>If FEAT_PAuth is implemented and TCR_EL1.TBID0 is 1, then this field only applies to Data accesses.</p> <p>Otherwise, if the value of TBIO is 1 and bit [55] of the target address to be stored to the PC is 0, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:</p> <ul style="list-style-type: none"> • A branch or procedure return within EL0 or EL1. • An exception taken to EL1. • An exception return to EL0 or EL1. <p>Otherwise</p> <p>RES0</p>	xxxx

Bits	Name	Description	Reset
[36]	AS	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>ASID Size.</p> <p>0b0</p> <p>8 bit - the upper 8 bits of AArch64-TTBRO_EL1 and AArch64-TTBR1_EL1 are ignored by hardware for every purpose except reading back the register, and are treated as if they are all zeros for when used for allocation and matching entries in the TLB.</p> <p>0b1</p> <p>16 bit - the upper 16 bits of AArch64-TTBRO_EL1 and AArch64-TTBR1_EL1 are used for allocation and matching in the TLB.</p> <p>When AArch64-VTCR_EL2.MSA == '0'</p> <p>ASID Size.</p> <p>0b0</p> <p>8 bit - the upper 8 bits of AArch64-TTBRO_EL1 are ignored by hardware for every purpose except reading back the register, and are treated as if they are all zeros when used for address matching.</p> <p>0b1</p> <p>16 bit - the upper 16 bits of AArch64-TTBRO_EL1 are used for address matching.</p> <p>Otherwise</p> <p>RES0</p>	xxxx
[35]	RES0	Reserved	RES0
[34:32]	IPS	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Intermediate Physical Address Size.</p> <p>0b000</p> <p>32 bits, 4GB.</p> <p>0b001</p> <p>36 bits, 64GB.</p> <p>0b010</p> <p>40 bits, 1TB.</p> <p>0b011</p> <p>42 bits, 4TB.</p> <p>0b100</p> <p>44 bits, 16TB.</p> <p>0b101</p> <p>48 bits, 256TB.</p> <p>0b110</p> <p>52 bits, 4PB.</p> <p>All other values are reserved.</p> <p>The reserved values behave in the same way as the 0b101 or 0b110 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.</p> <p>Otherwise</p> <p>RES0</p>	xxxx

Bits	Name	Description	Reset
[31:30]	TG1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Granule size for the AArch64-TTBR1_EL1.</p> <p>0b01 16KB.</p> <p>0b10 4KB.</p> <p>0b11 64KB.</p> <p>Other values are reserved.</p> <p>If the value is programmed to either a reserved value or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to be 0b10 for all purposes other than the value read back from this register.</p> <p>The value read back is the value programmed.</p> <p>Otherwise RES0</p>	xxxx
[29:28]	SH1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Shareability attribute for memory associated with translation table walks using AArch64-TTBR1_EL1.</p> <p>0b00 Non-shareable.</p> <p>0b10 Outer Shareable.</p> <p>0b11 Inner Shareable.</p> <p>Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.</p> <p>Otherwise RES0</p>	xxxx
[27:26]	ORGN1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Outer cacheability attribute for memory associated with translation table walks using AArch64-TTBR1_EL1.</p> <p>0b00 Normal memory, Outer Non-cacheable.</p> <p>0b01 Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.</p> <p>0b10 Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.</p> <p>0b11 Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.</p> <p>Otherwise RES0</p>	xxxx

Bits	Name	Description	Reset
[25:24]	IRGN1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Inner cacheability attribute for memory associated with translation table walks using AArch64-TTBR1_EL1.</p> <p>0b00 Normal memory, Inner Non-cacheable.</p> <p>0b01 Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable.</p> <p>0b10 Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable.</p> <p>0b11 Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable.</p> <p>Otherwise RES0</p>	xxxx
[23]	EPD1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Translation table walk disable for translations using AArch64-TTBR1_EL1. This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using AArch64-TTBR1_EL1. The encoding of this bit is:</p> <p>0b0 Perform translation table walks using AArch64-TTBR1_EL1.</p> <p>0b1 A TLB miss on an address that is translated using AArch64-TTBR1_EL1 generates a Translation fault. No translation table walk is performed.</p> <p>Otherwise RES0</p>	xxxx
[22]	A1	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Selects whether AArch64-TTBR0_EL1 or AArch64-TTBR1_EL1 defines the ASID. The encoding of this bit is:</p> <p>0b0 AArch64-TTBR0_EL1.ASID defines the ASID.</p> <p>0b1 AArch64-TTBR1_EL1.ASID defines the ASID.</p> <p>Otherwise RES0</p>	xxxx
[21:16]	T1SZ	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>The size offset of the memory region addressed by AArch64-TTBR1_EL1. The region size is $2^{(64-T1SZ)}$ bytes.</p> <p>The maximum and minimum possible values for T1SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.</p> <p>Otherwise RES0</p>	xxxx

Bits	Name	Description	Reset
[15:14]	TG0	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Granule size for the AArch64-TTBRO_EL1.</p> <p>0b00 4KB</p> <p>0b01 64KB</p> <p>0b10 16KB</p> <p>Other values are reserved.</p> <p>If the value is programmed to either a reserved value or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to be 0b00 for all purposes other than the value read back from this register.</p> <p>The value read back is the value programmed.</p> <p>Otherwise RES0</p>	xxxx
[13:12]	SH0	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Shareability attribute for memory associated with translation table walks using AArch64-TTBRO_EL1.</p> <p>0b00 Non-shareable</p> <p>0b10 Outer Shareable</p> <p>0b11 Inner Shareable</p> <p>Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE.</p> <p>Otherwise RES0</p>	xxxx
[11:10]	ORGN0	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Outer cacheability attribute for memory associated with translation table walks using AArch64-TTBRO_EL1.</p> <p>0b00 Normal memory, Outer Non-cacheable.</p> <p>0b01 Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable.</p> <p>0b10 Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable.</p> <p>0b11 Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable.</p> <p>Otherwise RES0</p>	xxxx

Bits	Name	Description	Reset
[9:8]	IRGNO	When AArch64-VTCR_EL2.MSA == '1' Inner cacheability attribute for memory associated with translation table walks using AArch64-TTBRO_EL1. 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. Otherwise RES0	xxxx
[7]	EPDO	When AArch64-VTCR_EL2.MSA == '1' Translation table walk disable for translations using AArch64-TTBRO_EL1. This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using AArch64-TTBRO_EL1. The encoding of this bit is: 0b0 Perform translation table walks using AArch64-TTBRO_EL1. 0b1 A TLB miss on an address that is translated using AArch64-TTBRO_EL1 generates a Translation fault. No translation table walk is performed. Otherwise RES0	xxxx
[6]	RES0	Reserved	RES0
[5:0]	TOSZ	When AArch64-VTCR_EL2.MSA == '1' The size offset of the memory region addressed by AArch64-TTBRO_EL1. The region size is $2^{(64-TOSZ)}$ bytes. The maximum and minimum possible values for TOSZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter. Otherwise RES0	xxxx

Access

MRS <Xt>, TCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b010

MSR TCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0000	0b010

Accessibility

MRS <Xt>, TCR_EL1


```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TCR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TCR_EL1;
```

MSR TCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    TCR_EL1 = X[t, 64];
```

A.2.2.8 APIAKeyLo_EL1, Pointer Authentication Key A for Instruction (bits[63:0])

Holds bits[63:0] of key A used for authentication of instruction pointer values.



Note

The term APIAKey_EL1 is used to describe the concatenation of AArch64-APIAKeyHi_EL1: AArch64-APIAKeyLo_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

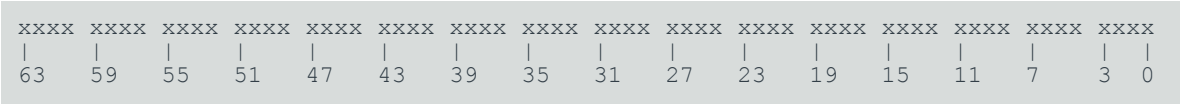
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-56: AARCH64_APIAKEYLO_EL1 bit assignments

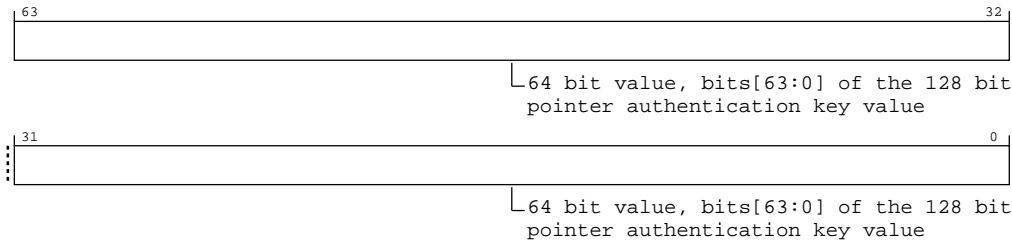


Table A-130: APIAKeyLo_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[63:0] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APIAKeyLo_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b000

MSR APIAKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b000

Accessibility

MRS <Xt>, APIAKeyLo_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APIAKeyLo_EL1;
    elseif PSTATE.EL == EL2 then
```

```
X[t, 64] = APIAKeyLo_EL1;
```

MSR APIAKeyLo_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APIAKeyLo_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    APIAKeyLo_EL1 = X[t, 64];
```

A.2.2.9 APIAKeyHi_EL1, Pointer Authentication Key A for Instruction (bits[127:64])

Holds bits[127:64] of key A used for authentication of instruction pointer values.



The term APIAKey_EL1 is used to describe the concatenation of AArch64-APIAKeyHi_EL1: AArch64-APIAKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-57: AARCH64_APIAKEYHI_EL1 bit assignments

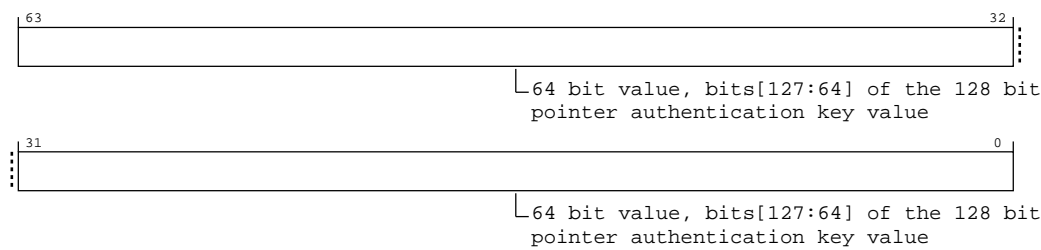


Table A-133: APIAKeyHi_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[127:64] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APIAKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b001

MSR APIAKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b001

Accessibility

MRS <Xt>, APIAKeyHi_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APIAKeyHi_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = APIAKeyHi_EL1;
```


MSR APIAKeyHi_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APIAKeyHi_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
```

```
APIAKeyHi_EL1 = X[t, 64];
```

A.2.2.10 APIBKeyLo_EL1, Pointer Authentication Key B for Instruction (bits[63:0])

Holds bits[63:0] of key B used for authentication of instruction pointer values.



Note

The term APIBKey_EL1 is used to describe the concatenation of AArch64-APIBKeyHi_EL1: AArch64-APIBKeyLo_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-58: AARCH64_APIBKEYLO_EL1 bit assignments

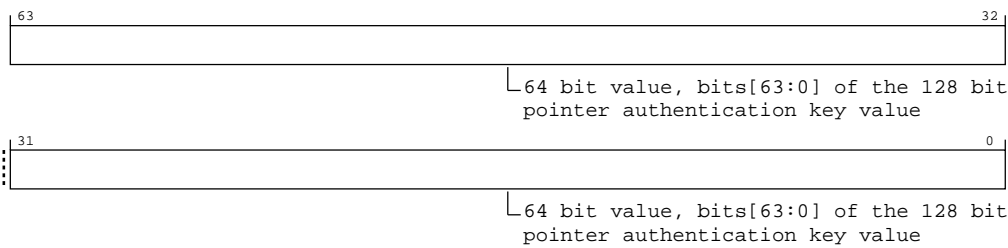


Table A-136: APIBKeyLo_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[63:0] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APIBKeyLo_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b010

MSR APIBKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b010

Accessibility

MRS <Xt>, APIBKeyLo_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APIBKeyLo_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = APIBKeyLo_EL1;

```

MSR APIBKeyLo_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APIBKeyLo_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    APIBKeyLo_EL1 = X[t, 64];

```

A.2.2.11 APIBKeyHi_EL1, Pointer Authentication Key B for Instruction (bits[127:64])

Holds bits[127:64] of key B used for authentication of instruction pointer values.

**Note**

The term APIBKey_EL1 is used to describe the concatenation of AArch64-APIBKeyHi_EL1: AArch64-APIBKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-59: AARCH64_APIBKEYHI_EL1 bit assignments

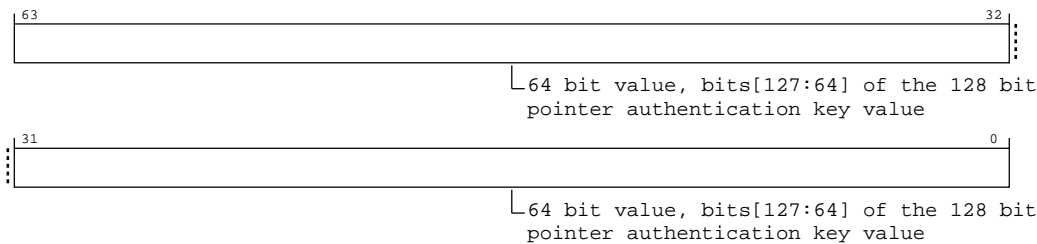


Table A-139: APIBKeyHi_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[127:64] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APIBKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b011

MSR APIBKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0001	0b011

Accessibility

MRS <Xt>, APIBKeyHi_EL1


```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APIBKeyHi_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = APIBKeyHi_EL1;
```

MSR APIBKeyHi_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APIBKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    APIBKeyHi_EL1 = X[t, 64];
```

A.2.2.12 APDAKeyLo_EL1, Pointer Authentication Key A for Data (bits[63:0])

Holds bits[63:0] of key A used for authentication of data pointer values.



Note

The term APDAKey_EL1 is used to describe the concatenation of AArch64-APDAKeyHi_EL1: AArch64-APDAKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-60: AARCH64_APDAKEYLO_EL1 bit assignments

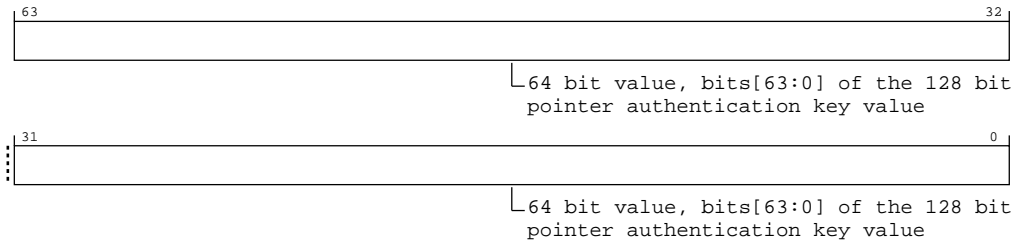


Table A-142: APDAKeyLo_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[63:0] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APDAKeyLo_EL1

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

MSR APDAKeyLo_EL1, <Xt>

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

Accessibility

MRS <Xt>, APDAKeyLo_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APDAKeyLo_EL1;
elseif PSTATE.EL == EL2 then
```

```
X[t, 64] = APDAKeyLo_EL1;
```

MSR APDAKeyLo_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APDAKeyLo_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    APDAKeyLo_EL1 = X[t, 64];
```

A.2.2.13 APDAKeyHi_EL1, Pointer Authentication Key A for Data (bits[127:64])

Holds bits[127:64] of key A used for authentication of data pointer values.



The term APDAKey_EL1 is used to describe the concatenation of AArch64-APDAKeyHi_EL1: AArch64-APDAKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-61: AARCH64_APDAKEYHI_EL1 bit assignments

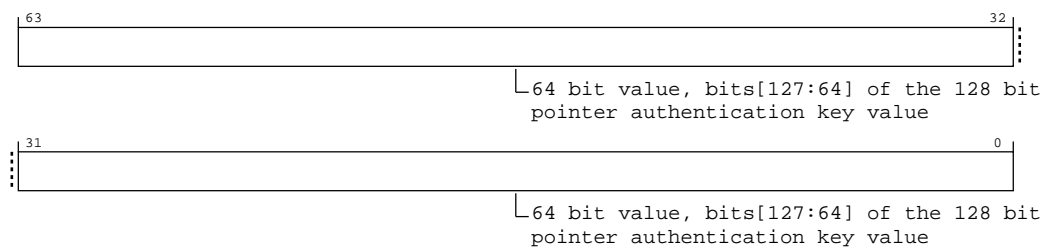


Table A-145: APDAKeyHi_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[127:64] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APDAKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b001

MSR APDAKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b001

Accessibility

MRS <Xt>, APDAKeyHi_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APDAKeyHi_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = APDAKeyHi_EL1;
```

MSR APDAKeyHi_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APDAKeyHi_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
```

```
APDAKeyHi_EL1 = X[t, 64];
```

A.2.2.14 APDBKeyLo_EL1, Pointer Authentication Key B for Data (bits[63:0])

Holds bits[63:0] of key B used for authentication of data pointer values.



The term APDBKey_EL1 is used to describe the concatenation of AArch64-APDBKeyHi_EL1: AArch64-APDBKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-62: AARCH64_APDBKEYLO_EL1 bit assignments

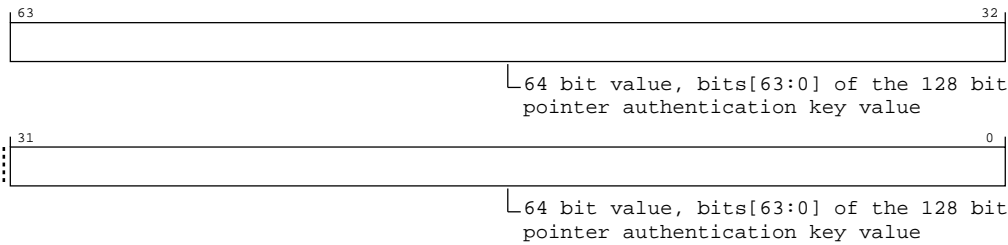


Table A-148: APDBKeyLo_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[63:0] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APDBKeyLo_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b010

MSR APDBKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b010

Accessibility

MRS <Xt>, APDBKeyLo_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APDBKeyLo_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = APDBKeyLo_EL1;

```

MSR APDBKeyLo_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APDBKeyLo_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    APDBKeyLo_EL1 = X[t, 64];

```

A.2.2.15 APDBKeyHi_EL1, Pointer Authentication Key B for Data (bits[127:64])

Holds bits[127:64] of key B used for authentication of data pointer values.

**Note**

The term APDBKey_EL1 is used to describe the concatenation of AArch64-APDBKeyHi_EL1: AArch64-APDBKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-63: AARCH64_APDBKEYHI_EL1 bit assignments

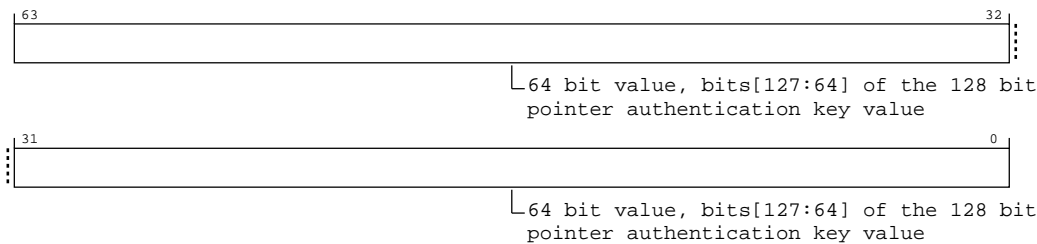


Table A-151: APDBKeyHi_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[127:64] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APDBKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b011

MSR APDBKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0010	0b011

Accessibility

MRS <Xt>, APDBKeyHi_EL1


```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APDBKeyHi_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = APDBKeyHi_EL1;
```

MSR APDBKeyHi_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APDBKeyHi_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    APDBKeyHi_EL1 = X[t, 64];
```

A.2.2.16 APGAKeyLo_EL1, Pointer Authentication Key A for Code (bits[63:0])

Holds bits[63:0] of key used for generic pointer authentication code.



Note

The term APGAKey_EL1 is used to describe the concatenation of AArch64-APGAKeyHi_EL1: AArch64-APGAKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-64: AARCH64_APGAKEYLO_EL1 bit assignments

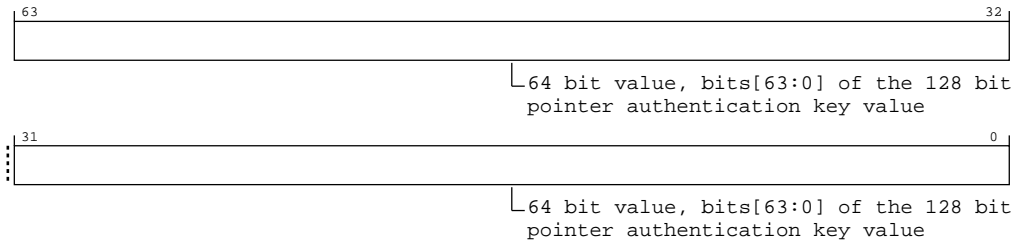


Table A-154: APGAKeyLo_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[63:0] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APGAKeyLo_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b000

MSR APGAKeyLo_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b000

Accessibility

MRS <Xt>, APGAKeyLo_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APGAKeyLo_EL1;
    elseif PSTATE.EL == EL2 then
```



```
X[t, 64] = APGAKeyLo_EL1;
```

MSR APGAKeyLo_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APGAKeyLo_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    APGAKeyLo_EL1 = X[t, 64];
```

A.2.2.17 APGAKeyHi_EL1, Pointer Authentication Key A for Code (bits[127:64])

Holds bits[127:64] of key used for generic pointer authentication code.



The term APGAKey_EL1 is used to describe the concatenation of AArch64-APGAKeyHi_EL1: AArch64-APGAKeyLo_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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-65: AARCH64_APGAKEYHI_EL1 bit assignments

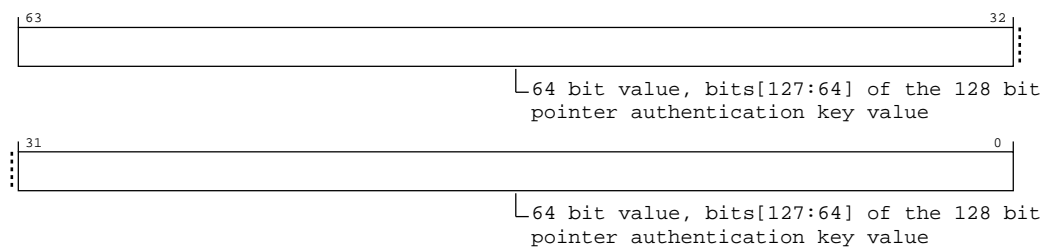


Table A-157: APGAKeyHi_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	64 bit value, bits[127:64] of the 128 bit pointer authentication key value.	64 {x}

Access

MRS <Xt>, APGAKeyHi_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b001

MSR APGAKeyHi_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0010	0b0011	0b001

Accessibility

MRS <Xt>, APGAKeyHi_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = APGAKeyHi_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = APGAKeyHi_EL1;
```

MSR APGAKeyHi_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.APK == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        APGAKeyHi_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
```

```
APGAKeyHi_EL1 = X[t, 64];
```

A.2.2.18 SPSel, Stack Pointer Select

Allows the Stack Pointer to be selected between SP_ELO and SP_ELx.

Configurations

This register is available in all configurations.

Attributes

Width

64

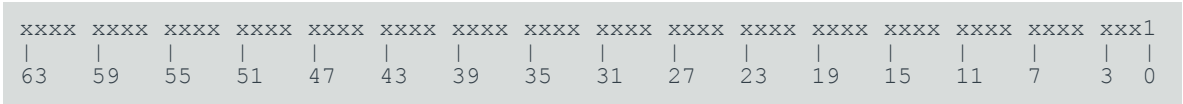
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-66: AARCH64_SPSEL bit assignments

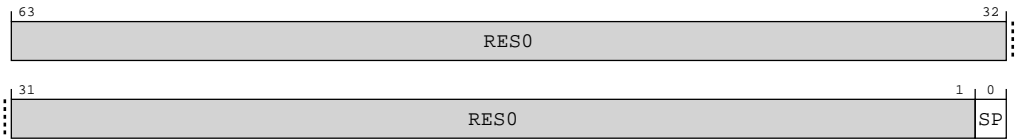


Table A-160: SPSel bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	SP	Stack pointer to use. Possible values of this bit are: 0b0 Use SP_ELO at all Exception levels. 0b1 Use SP_ELx for Exception level ELx.	0b1

Access

MRS <Xt>, SPSel

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b000

MSR SPSel, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b000

MSR SPSel, #<imm>

op0	op1	CRn	op2
0b00	0b000	0b0100	0b101

Accessibility

MRS <Xt>, SPSel

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(63):PSTATE.SP;
elsif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(63):PSTATE.SP;
```

MSR SPSel, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    PSTATE.SP = X[t, 64]<0>;
elsif PSTATE.EL == EL2 then
    PSTATE.SP = X[t, 64]<0>;
```

MSR SPSel, #<imm>

A.2.2.19 CurrentEL, Current Exception Level

Holds the current Exception level.

Configurations

This register is available in all configurations.

Attributes

Width

64

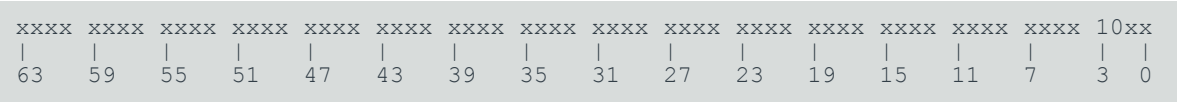
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-67: AARCH64_CURRENTEL bit assignments

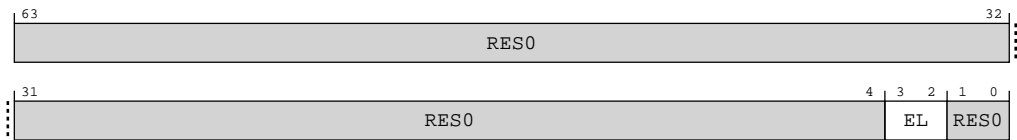


Table A-164: CurrentEL bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:2]	EL	Current Exception level. 0b00 EL0. 0b01 EL1. 0b10 EL2.	0b10

Bits	Name	Description	Reset
[1:0]	RES0	Reserved	RES0

Access

MRS <Xt>, CurrentEL

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b010

Accessibility

MRS <Xt>, CurrentEL

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(60):PSTATE.EL:Zeros(2);
elsif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(60):PSTATE.EL:Zeros(2);

```

A.2.2.20 PAN, Privileged Access Never

Allows access to the Privileged Access Never bit.

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-68: AARCH64_PAN bit assignments

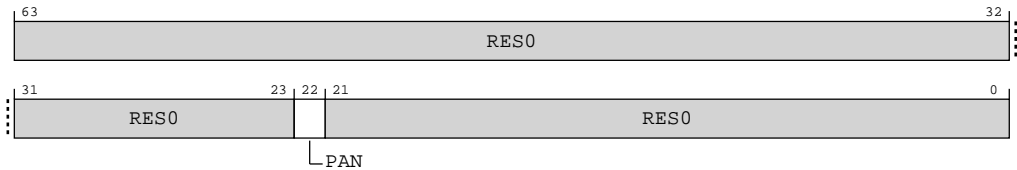


Table A-166: PAN bit descriptions

Bits	Name	Description	Reset
[63:23]	RES0	Reserved	RES0
[22]	PAN	Privileged Access Never. 0b0 Privileged reads and write are not disabled by this mechanism. 0b1 Disables privileged read and write accesses to addresses accessible at EL0 for an enabled stage 1 translation regime that defines the EL0 permissions. The value of this bit is usually preserved on taking an exception, except in the following situations: <ul style="list-style-type: none">When the target of the exception is EL1, and the value of the AArch64-SCTLR_EL1.SPAN bit is 0, this bit is set to 1.When the target of the exception is EL2, AArch64-HCR_EL2.{E2H, TGE} is {1, 1}, and the value of the AArch64-SCTLR_EL2.SPAN bit is 0, this bit is set to 1.	x
[21:0]	RES0	Reserved	RES0

Access

MRS <Xt>, PAN

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b011

MSR PAN, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b011

MSR PAN, #<imm>

op0	op1	CRn	op2
0b00	0b000	0b0100	0b100

Accessibility

MRS <Xt>, PAN

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(41):PSTATE.PAN:Zeros(22);
elseif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(41):PSTATE.PAN:Zeros(22);
```

MSR PAN, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    PSTATE.PAN = X[t, 64]<22>;
elseif PSTATE.EL == EL2 then
    PSTATE.PAN = X[t, 64]<22>;
```

MSR PAN, #<imm>

A.2.2.21 UAO, User Access Override

Allows access to the User Access Override bit.

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-69: AARCH64_UAO bit assignments

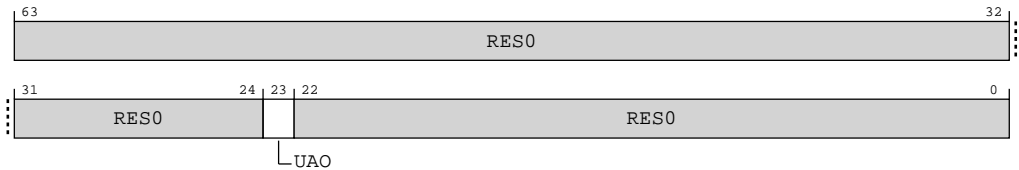


Table A-170: UAO bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23]	UAO	User Access Override. 0b0 The behavior of LDTR* and STTR* instructions is as defined in the base Armv8 architecture. 0b1 When executed at the following Exception levels, LDTR* and STTR* instructions behave as the equivalent LDR* and STR* instructions: <ul style="list-style-type: none">EL1. When executed at EL2, the LDTR* and STTR* instructions behave as the equivalent LDR* and STR* instructions, regardless of the setting of the PSTATE.UAO bit.	x
[22:0]	RES0	Reserved	RES0

Access

For more information about the operation of the MSR (immediate) accessor, see *MSR (immediate)* in the *Arm® Architecture Reference Manual for A-profile architecture*.

MRS <Xt>, UAO

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b100

MSR UAO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b100

MSR UAO, #<imm>

op0	op1	CRn	op2
0b00	0b000	0b0100	0b011

Accessibility

For more information about the operation of the MSR (immediate) accessor, see 'MSR (immediate)'.

MRS <Xt>, UAO

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(40):PSTATE.UAO:Zeros(23);
elseif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(40):PSTATE.UAO:Zeros(23);
```

MSR UAO, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    PSTATE.UAO = X[t, 64]<23>;
elseif PSTATE.EL == EL2 then
    PSTATE.UAO = X[t, 64]<23>;
```

MSR UAO, #<imm>

A.2.2.22 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1)

Provides additional implementation defined fault status information for Data Abort, Instruction Abort or SError interrupt 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-70: AARCH64_AFSR0_EL1 bit assignments

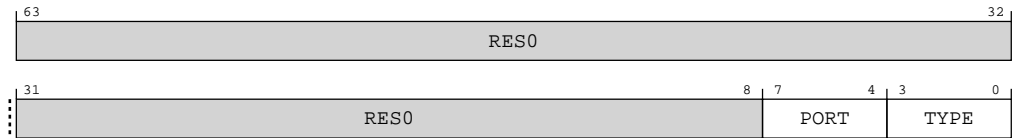


Table A-174: AFSR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:4]	PORT	<p>Memory or bus interface that caused the abort.</p> <p>0b0000 MM. Main Manager port.</p> <p>0b0001 LLRAM. Low-Latency RAM port.</p> <p>0b0010 LLPP. Low-Latency Peripheral Port.</p> <p>0b0011 SPP. Shared Peripheral Port.</p> <p>0b0100 ITCM. Instruction Tightly Coupled Memory.</p> <p>0b0101 DTCM. Data Tightly Coupled Memory.</p> <p>0b0110 Overlapping or illegal port. Used for accesses that target multiple ports, when they simultaneously belong in two or more of the ITCM, DTCM, LLRAM, LLPP or SPP enabled regions. Also used for page table accesses if they belong in any of the ITCM, DTCM, LLRAM, LLPP or SPP enabled regions.</p> <p>0b0111 Unknown. All faults other than Instruction or Data aborts will have this type. For External aborts and Unsupported Exclusive or Atomic access faults, this is used for ambiguous memory accesses or for faults that do not have an associated address. For other Instruction or Data aborts, this indicates that the port information is not applicable to the fault.</p> <p>Other values are Reserved.</p>	xxxx
[3:0]	TYPE	<p>Fault type.</p> <p>0b0000 Error on data.</p> <p>0b0010 Error on interrupt latency. Indicates the abort was generated by an instruction trapped by AArch64-IMP_INTLATENCY_EL2 controls.</p> <p>0b0011 Other error.</p> <p>Other values are Reserved.</p>	xxxx

Access

MRS <Xt>, AFSRO_EL1

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

MSR AFSRO_EL1, <Xt>

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

Accessibility

MRS <Xt>, AFSRO_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = AFSRO_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = AFSRO_EL1;
```

MSR AFSRO_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSRO_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    AFSRO_EL1 = X[t, 64];
```

A.2.2.23 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

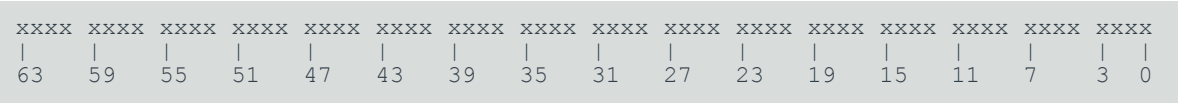
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-71: AARCH64_AFSR1_EL1 bit assignments

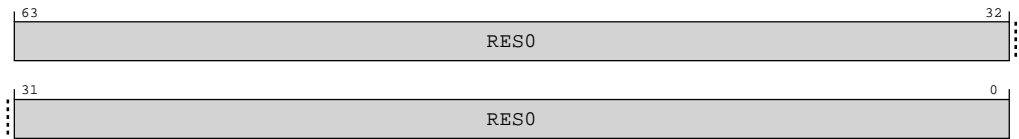


Table A-177: AFSR1_EL1 bit descriptions

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

Access

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

MRS <Xt>, AFSR1_EL1

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

```
X[t, 64] = AFSR1_EL1;
```

MSR AFSR1_EL1, <Xt>

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

A.2.2.24 ESR_EL1, Exception Syndrome Register (EL1)

Holds syndrome information for an exception 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	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

ESR_EL1 is made **UNKNOWN** as a result of an exception return from EL1.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL1, the value of ESR_EL1 is **UNKNOWN**. The value written to ESR_EL1 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Figure A-72: AARCH64_ESR_EL1 bit assignments

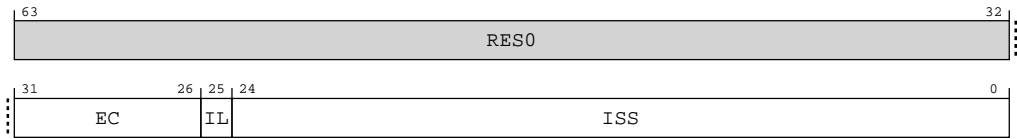


Table A-180: ESR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:26]	EC	<p>Exception Class. Indicates the reason for the exception that this register holds information about.</p> <p>For each EC value, the table references a subsection that gives information about:</p> <ul style="list-style-type: none">The cause of the exception, for example the configuration required to enable the trap.The encoding of the associated ISS. <p>Possible values of the EC field are:</p> <p>0b000000 Unknown reason.</p> <p>0b000001 Trapped WFI or WFE instruction execution.</p> <p>Conditional WFE and WFI instructions that fail their condition code check do not cause an exception.</p> <p>0b000111 Access to Advanced SIMD, or floating-point functionality trapped by AArch64-CPACR_EL1.FPEN, AArch64-CPTR_EL2.FPEN, or AArch64-CPTR_EL2.TFP control.</p> <p>Excludes exceptions resulting from AArch64-CPACR_EL1 when the value of AArch64-HCR_EL2.TGE is 1, or because Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000.</p> <p>0b001110 Illegal Execution state.</p>	6 {x}

Bits	Name	Description	Reset
[31:26 continued]	EC	<p>0b010101 SVC instruction execution in AArch64 state.</p> <p>0b011000 Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111.</p> <p>This includes all instructions that cause exceptions that are part of the encoding space defined 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.</p> <p>0b011100 Exception from a Pointer Authentication instruction authentication failure</p> <p>0b100000 Instruction Abort from a lower Exception level.</p> <p>Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p>	6{x}
[31:26 continued]	EC	<p>0b100001 Instruction Abort taken without a change in Exception level.</p> <p>Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p> <p>0b100010 PC alignment fault exception.</p> <p>0b100100 Data Abort from a lower Exception level.</p> <p>Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p> <p>0b100101 Data Abort taken without a change in Exception level.</p> <p>Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p> <p>0b100110 SP alignment fault exception.</p> <p>0b101111 SError interrupt.</p> <p>0b110000 Breakpoint exception from a lower Exception level.</p>	6{x}

Bits	Name	Description	Reset
[31:26 continued]	EC	<p>0b110001 Breakpoint exception taken without a change in Exception level.</p> <p>0b110010 Software Step exception from a lower Exception level.</p> <p>0b110011 Software Step exception taken without a change in Exception level.</p> <p>0b110100 Watchpoint exception from a lower Exception level.</p> <p>0b110101 Watchpoint exception taken without a change in Exception level.</p> <p>0b111100 BRK instruction execution in AArch64 state.</p> <p>All other EC values are reserved by Arm, and:</p> <ul style="list-style-type: none"> Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions. Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions. <p>The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE.</p>	6 {x}
[25]	IL	<p>Instruction Length for synchronous exceptions. Possible values of this bit are:</p> <p>0b0 16-bit instruction trapped.</p> <p>0b1 32-bit instruction trapped. This value is also used when the exception is one of the following:</p> <ul style="list-style-type: none"> An SError interrupt. An Instruction Abort exception. A PC alignment fault exception. An SP alignment fault exception. A Data Abort exception for which the value of the ISV bit is 0. An Illegal Execution state exception. Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning: <ul style="list-style-type: none"> 0b1: 32-bit A64 BRK instruction. An exception reported using EC value 0b000000. 	x
[24:0]	ISS	<p>ISS encoding for exceptions with an unknown reason</p> <p>24:0 Reserved, RES0.</p> <p>When an exception is reported using this EC code the IL field is set to 1.</p> <p>This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:</p>	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	<ul style="list-style-type: none"> The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including: <ul style="list-style-type: none"> A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state. A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state. Instruction encodings that are unallocated. Instruction encodings for instructions or System registers that are not implemented in the implementation. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state. In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers. Attempted execution of: <ul style="list-style-type: none"> An HVC instruction when disabled by AArch64-HCR_EL2.HCD. An SMC instruction. An HLT instruction when disabled by ext-EDSCR.HDE. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> Attempted execution of an MSR or MRS instruction to access AArch64-SP_ELO when the value of AArch64-SPSel.SP is 0. Attempted execution of an MSR or MRS instruction using a _EL12 register name. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> Attempted execution, in Debug state, of: <ul style="list-style-type: none"> A DCPS1 instruction when the value of AArch64-HCR_EL2.TGE is 1 and EL2 is disabled or not implemented in the current Security state. A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state. A DCPS3 instruction 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> An exception that is taken to EL2 because the value of AArch64-HCR_EL2.TGE is 1 that, if the value of AArch64-HCR_EL2.TGE was 0 would have been reported with an ESR_ELx.EC value of 0b000111. 	25 {x}
[24:0 continued]	ISS	<p>ISS encoding for an exception from a WF* instruction</p> <p>24</p> <p>Condition code valid.</p> <p>0b0</p> <p>The COND field is not valid.</p> <p>0b1</p> <p>The COND field is valid.</p> <p>23:20</p> <p>For exceptions taken from AArch64, this field is set to 0b1110.</p>	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	19:2 Reserved, RES0 . 1:0 Trapped instruction. Possible values of this bit are: 0b00 WFI trapped. 0b01 WFE trapped. The following fields describe configuration settings for generating this exception: <ul style="list-style-type: none"> AArch64-SCTLR_EL1.{nTWE, nTWI}. AArch64-HCR_EL2.{TWE, TWI}. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps The accesses covered by this trap include: <ul style="list-style-type: none"> Execution of SVE or Advanced SIMD and floating-point instructions. Accesses to the Advanced SIMD and floating-point System registers. For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.	25 {x}
[24:0 continued]	ISS	24 Condition code valid. 0b0 The COND field is not valid. 0b1 The COND field is valid.	25 {x}
[24:0 continued]	ISS	23:20 For exceptions taken from AArch64, this field is set to 0b1110. 19:0 Reserved, RES0 . The following fields describe the configuration settings for the traps that are reported using EC value 0b000111: <ul style="list-style-type: none"> AArch64-CPACR_EL1.FPEN, for accesses to SIMD and floating-point registers trapped to EL1. AArch64-CPTR_EL2.FPEN and AArch64-CPTR_EL2.TFP, for accesses to SIMD and floating-point registers trapped to EL2. 	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault 24:0 Reserved, RES0 . <p>There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about PC alignment fault exceptions, see <i>PC alignment checking</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p><i>SP alignment checking</i> in the Arm® Architecture Reference Manual for A-profile architecture describes the configuration settings for generating SP alignment fault exceptions.</p>	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from HVC or SVC instruction execution 24:16 Reserved, RES0 . <p>For A64 instructions, see SVC in the and 'HVC'.</p>	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state 24:22 Reserved, RES0 . 21:20 The Op0 value from the issued instruction. 19:17 The Op2 value from the issued instruction.	25 {x}
[24:0 continued]	ISS	16:14 The Op1 value from the issued instruction. 13:10 The CRn value from the issued instruction. 9:5 The Rt value from the issued instruction, the general-purpose register used for the transfer.	25 {x}
[24:0 continued]	ISS	4:1 The CRm value from the issued instruction. 0 Indicates the direction of the trapped instruction. 0b0 Write access, including MSR instructions. 0b1 Read access, including MRS instructions	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	<p>For exceptions caused by System instructions, see <i>System instructions' subsection of 'Branches, exception generating and System instructions</i> in the Arm® Architecture Reference Manual for A-profile architecture for the encoding values returned by an instruction.</p> <p>The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:</p> <ul style="list-style-type: none"> AArch64-SCTLR_EL1.UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-SCTLR_EL1.UCT, for accesses to AArch64-CTR_EL0 using AArch64 state, MSR or MRS access trapped to EL1 or EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-SCTLR_EL1.DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-SCTLR_EL1.UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-CPACR_EL1.TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-CNTKCTL_EL1.{ELOPTEN, ELOVTEN, ELOPCTEN, ELOVCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-HCR_EL2.TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.TACR, for accesses to the Auxiliary Control Register, AArch64-ACTLR_EL1, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-HCR_EL2.{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-CPTR_EL2.TCPAC, for accesses to AArch64-CPACR_EL1, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-CPTR_EL2.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-MDCR_EL2.TTRF, for accesses to the trace filter control register, AArch64-TRFCR_EL1, using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-MDCR_EL2.TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-MDCR_EL2.TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2. AArch64-CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2. AArch64-MDCR_EL2.TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2. AArch64-HCR_EL2.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from an Instruction Abort 24:13 Reserved, RES0 .	25 {x}
[24:0 continued]	ISS	12:11 Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception. 0b00 Recoverable state (UER). 0b10 Uncontainable (UC). 0b11 Restartable state (UEO).	25 {x}
[24:0 continued]	ISS	10 FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk. 0b0 FAR is valid. 0b1 FAR is not valid, and holds an UNKNOWN value. 9 External abort type. 0b0 No IMPLEMENTATION DEFINED classification of External aborts.	25 {x}
[24:0 continued]	ISS	8 Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction: 0b0 The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	0b1 The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The r[DC ZVA](AArch64-dc-zva), r[DC GVA](AArch64-dc-gva), and r[DC GZVA](AArch64-dc-gzva) instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.	25 {x}
[24:0 continued]	ISS	7 For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk: 0b0 Fault not on a stage 2 translation for a stage 1 translation table walk. 0b1 Fault on the stage 2 translation of an access for a stage 1 translation table walk.	25 {x}
[24:0 continued]	ISS	6 Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location. 0b0 Abort caused by an instruction reading from a memory location. 0b1 Abort caused by an instruction writing to a memory location.	25 {x}
[24:0 continued]	ISS	5:0 Data Fault Status Code. 0b000000 Address size fault, level 0 of translation or translation table base register. 0b000001 Address size fault, level 1.	25 {x}
[24:0 continued]	ISS	0b000010 Address size fault, level 2. 0b000011 Address size fault, level 3. 0b000100 Translation fault, level 0. 0b000101 Translation fault, level 1. 0b000110 Translation fault, level 2. 0b000111 Translation fault, level 3.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	0b001001 Access flag fault, level 1. 0b001010 Access flag fault, level 2. 0b001011 Access flag fault, level 3. 0b001100 Permission fault, level 0. 0b001101 Permission fault, level 1.	25 {x}
[24:0 continued]	ISS	0b001110 Permission fault, level 2. 0b001111 Permission fault, level 3. 0b010000 Synchronous External abort, not on translation table walk or hardware update of translation table. 0b010100 Synchronous External abort on translation table walk or hardware update of translation table, level 0.	25 {x}
[24:0 continued]	ISS	0b010101 Synchronous External abort on translation table walk or hardware update of translation table, level 1. 0b010110 Synchronous External abort on translation table walk or hardware update of translation table, level 2. 0b010111 Synchronous External abort on translation table walk or hardware update of translation table, level 3.	25 {x}
[24:0 continued]	ISS	0b100001 Alignment fault. 0b110000 TLB conflict abort. 0b110001 Unsupported atomic hardware update fault. 0b110100 IMPLEMENTATION DEFINED fault (Lockdown). 0b110101 IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	25 {x}
[24:0 continued]	ISS	ISS encoding for an SError interrupt 24 IMPLEMENTATION DEFINED syndrome. 0b0 Bits [23:0] of the ISS field holds the fields described in this encoding. 0b1 Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	23:14 Reserved, RES0 . 13 Implicit error synchronization event. 0b0 The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately. 0b1 The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.	25 {x}
[24:0 continued]	ISS	12:10 Asynchronous Error Type. When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception. 0b000 Uncontainable (UC). 0b001 Unrecoverable state (UEU). 0b010 Restartable state (UEO). 0b011 Recoverable state (UER). 0b110 Corrected (CE).	25 {x}
[24:0 continued]	ISS	9 External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts. This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors. 8:6 Reserved, RES0 . 5:0 Data Fault Status Code. 0b000000 Uncategorized error. 0b010001 Asynchronous SError interrupt.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	ISS encoding for an exception from a Breakpoint or Vector Catch debug exception 24:6 Reserved, RES0. 5:0 Instruction Fault Status Code. 0b100010 Debug exception. For more information about generating these exceptions: <ul style="list-style-type: none"> For exceptions from AArch64, see <i>Breakpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from a Software Step exception 24 Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows: 0b0 EX bit is RES0. 0b1 EX bit is valid. 23:7 Reserved, RES0.	25 {x}
[24:0 continued]	ISS	6 Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped. 0b0 An instruction other than a Load-Exclusive instruction was stepped. 0b1 A Load-Exclusive instruction was stepped. 5:0 Instruction Fault Status Code. 0b100010 Debug exception. For more information about generating these exceptions, see <i>Software Step exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture .	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from a Watchpoint exception 24:9 Reserved, RES0.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	8 Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation instruction: 0b0 The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1. 0b1 The Watchpoint exception was generated by either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation instruction. The r[DC ZVA](AArch64-dc-zva), r[DC GVA](AArch64-dc-gva), and r[DC GZVA](AArch64-dc-gzva) instructions are not classified as a cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1	25 {x}
[24:0 continued]	ISS	7 Reserved, RES0 . 6 Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location. 0b0 Watchpoint exception caused by an instruction reading from a memory location. 0b1 Watchpoint exception caused by an instruction writing to a memory location. 5:0 Data Fault Status Code. 0b100010 Debug exception. For more information about generating these exceptions, see <i>Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture .	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from execution of a Breakpoint instruction 24:16 Reserved, RES0 . 15:0 Set to the instruction comment field value, zero extended as necessary. For more information about generating these exceptions, see <i>Breakpoint instruction exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture .	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	ISS encoding for an exception from a Pointer Authentication instruction authentication failure 24:2 Reserved, RES0 . 1 This field indicates whether the exception is as a result of an Instruction key or a Data key. 0b0 Instruction Key. 0b1 Data Key. 0 This field indicates whether the exception is as a result of an A key or a B key. 0b0 A key. 0b1 B key	25 {x}
[24:0 continued]	ISS	The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect: <ul style="list-style-type: none"> AUTIASP, AUTIAZ, AUTIA1716. AUTIBSP, AUTIBZ, AUTIB1716. AUTIA, AUTDA, AUTIB, AUTDB. AUTIZA, AUTIZB, AUTDZA, AUTDZB. RETAA, RETAB. BRAA, BRAB, BLRAA, BLRAB. BRAAZ, BRABZ, BLRAAZ, BLRABZ. ERETAA, ERETAB. LDRAA, LDRAB, whether the authenticated address is written back to the base register or not. 	25 {x}

Access

MRS <Xt>, ESR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

MSR ESR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

Accessibility

MRS <Xt>, ESR_EL1

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

```
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ESR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ESR_EL1;
```

MSR ESR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ESR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ESR_EL1 = X[t, 64];
```

A.2.2.25 PRBAR<n>_EL1, Protection Region Base Address Register n (EL1), n = 1 - 15

Provides access to the base address for the MPU region determined by the value of 'n' and AArch64-PRSELR_EL1.REGION as AArch64-PRSELR_EL1.REGION<7:4>:n.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-73: AARCH64_PRBAR<n>_EL1 bit assignments

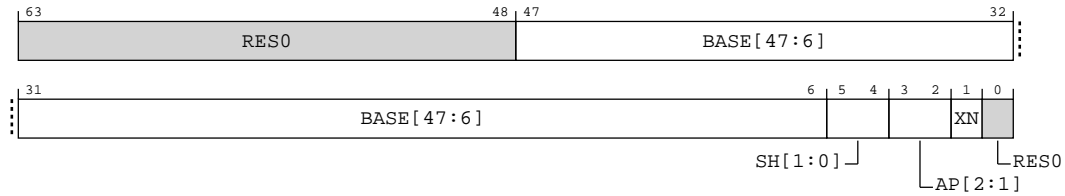


Table A-183: PRBAR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:6]	BASE[47:6]	Bits[47:6] of the lower inclusive limit of the selected EL1 MPU memory region. This value is zero extended to provide the base address to be checked against.	42 {x}
[5:4]	SH[1:0]	Shareability attribute. 0b00 Non-shareable 0b01 Reserved, CONSTRAINED UNPREDICTABLE 0b10 Outer Shareable 0b11 Inner Shareable	xx
[3:2]	AP[2:1]	Access Permission attributes. 0b00 Read/write at EL1, no access at ELO 0b01 Read/write at EL1 and ELO 0b10 Read-only at EL1, no access at ELO 0b11 Read-only at EL1 and ELO	xx
[1]	XN	Execute Never 0b0 Execution of instructions fetched from the region is permitted. 0b1 Execution of instructions fetched from the region is not permitted.	x
[0]	RES0	Reserved	RES0

Access

Any access to MPU region register PRBAR<n>_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR<n>_EL1 return an **UNKNOWN** value.
- Writes to unimplemented PRBAR<n>_EL1 register make all PRBAR_EL1 registers value **UNKNOWN**.

MRS <Xt>, PRBAR<m>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	'1':m[3:1]	m[0]:'00'

MSR PRBAR<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	'1':m[3:1]	m[0]:'00'

Accessibility

Any access to MPU region register PRBAR<n>_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR<n>_EL1 return an **UNKNOWN** value.
- Writes to unimplemented PRBAR<n>_EL1 register make all PRBAR_EL1 registers value **UNKNOWN**.

MRS <Xt>, PRBAR<m>_EL1

```
integer m = UInt(CRm<2:0>:op2<2>);

if m + (UInt(PRSELR_EL1.REGION<7:4>) * 16) >= UInt(MPUIR_EL1.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        X[t, 64] = PRBAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)];
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRBAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)];
```

MSR PRBAR<m>_EL1, <Xt>

```
integer m = UInt(CRm<2:0>:op2<2>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
```

```
else
    PRBAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)] = X[t, 64];
elseif PSTATE.EL == EL2 then
    PRBAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)] = X[t, 64];
```

A.2.2.26 PRLAR<n>_EL1, Protection Region Limit Address Register n (EL1), n = 1 - 15

Provides access to the limit address for the MPU region determined by the value of 'n' and AArch64-PRSELR_EL1.REGION as AArch64-PRSELR_EL1.REGION<7:4>:n.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

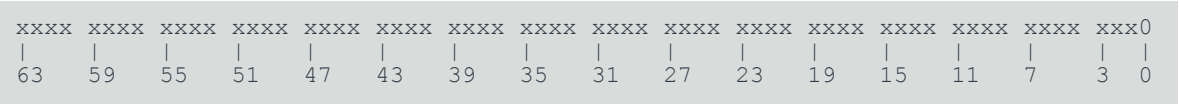
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-74: AARCH64_PRLAR<n>_EL1 bit assignments

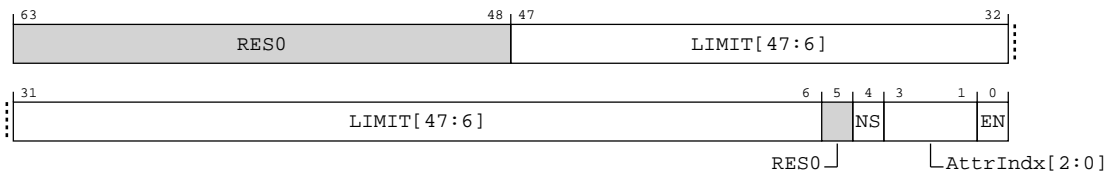


Table A-186: PRLAR<n>_EL1 bit descriptions

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

Bits	Name	Description	Reset
[47:6]	LIMIT[47:6]	Bits[47:6] of the upper inclusive limit of the selected EL1 MPU memory region. This value is concatenated with the value 0x3F to provide the limit address to be checked against.	42 {x}
[5]	RES0	Reserved	RES0
[4]	NS	Non-secure bit. Specifies whether the output address is in the Secure or Non-secure memory. 0b0 Output address is in Secure address space. 0b1 Output address is in Non-secure address space.	x
[3:1]	AttrIdx[2:0]	Selects attributes from within the associated Memory Attribute Indirection Register. 0b000 Select the Attr0 field from MAIR_EL1. 0b001 Select the Attr1 field from MAIR_EL1. 0b010 Select the Attr2 field from MAIR_EL1. 0b011 Select the Attr3 field from MAIR_EL1. 0b100 Select the Attr4 field from MAIR_EL1. 0b101 Select the Attr5 field from MAIR_EL1. 0b110 Select the Attr6 field from MAIR_EL1. 0b111 Select the Attr7 field from MAIR_EL1.	xxx
[0]	EN	Region enable bit. 0b0 Region disabled. 0b1 Region enabled.	0b0

Access

Any access to MPU region register PRLAR<n>_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR<n>_EL1 return an **UNKNOWN** value.
- Writes to unimplemented PRLAR<n>_EL1 register make all PRLAR_EL1 registers value **UNKNOWN**.

MRS <Xt>, PRLAR<m>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	'1':m[3:1]	m[0]:'01'

MSR PRLAR<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	'1':m[3:1]	m[0]:'01'

Accessibility

Any access to MPU region register PRLAR<n>_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is CONSTRAINED UNPREDICTABLE.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR<n>_EL1 return an UNKNOWN value.
- Writes to unimplemented PRLAR<n>_EL1 register make all PRLAR_EL1 registers value UNKNOWN.

MRS <Xt>, PRLAR<m>_EL1

```
integer m = UInt(CRm<2:0>:op2<2>);
if (UInt(PRSELR_EL1.REGION<7:4>) * 16) + m >= UInt(MPUIR_EL1.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        X[t, 64] = PRLAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)];
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRLAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)];
```

MSR PRLAR<m>_EL1, <Xt>

```
integer m = UInt(CRm<2:0>:op2<2>);
if (UInt(PRSELR_EL1.REGION<7:4>) * 16) + m >= UInt(MPUIR_EL1.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        PRLAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)] = X[t, 64];
elseif PSTATE.EL == EL2 then
    PRLAR_EL1[m + (UInt(AArch64-PRSELR_EL1.REGION<7:4>) * 16)] = X[t, 64];
```

A.2.2.27 FAR_EL1, Fault Address Register (EL1)

Holds the faulting Virtual Address for all synchronous Instruction Abort exceptions, Data Abort exceptions, PC alignment fault exceptions and Watchpoint exceptions that are 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-75: AARCH64_FAR_EL1 bit assignments

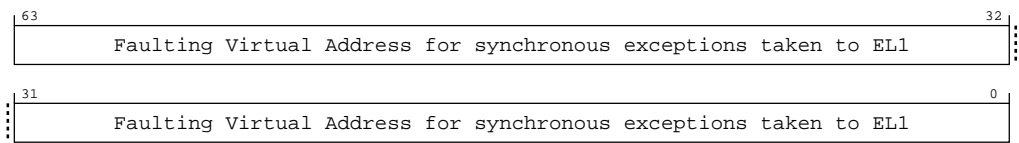


Table A-189: FAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	<p>Faulting Virtual Address for synchronous exceptions taken to EL1. Exceptions that set the FAR_EL1 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), PC alignment faults (EC 0x22), and Watchpoints (EC 0x34 or 0x35). AArch64-ESR_EL1.EC holds the EC value for the exception.</p> <p>For a synchronous External abort, if the VA that generated the abort was from an address range for which $\text{TCR_ELx.TBI}\{\langle 0 1 \rangle\} == 1$ for the translation regime in use when the abort was generated, then the top eight bits of FAR_EL1 are UNKNOWN.</p> <p>For a synchronous External abort other than a synchronous External abort on a translation table walk, this field is valid only if AArch64-ESR_EL1.FnV is 0, and FAR_EL1 is UNKNOWN if AArch64-ESR_EL1.FnV is 1.</p> <p>If a memory fault that sets FAR_EL1, other than a Tag Check Fault, is generated from a data cache maintenance or other DC instruction, this field holds the address specified in the register argument of the instruction.</p> <p>For a Data Abort exception or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see <i>Address tagging</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>Execution at EL0 makes FAR_EL1 become UNKNOWN.</p> <p>Note: The address held in this field is an address accessed by the instruction fetch or data access that caused the exception that actually gave rise to the instruction or data abort. It is the lower address that gave rise to the fault that is reported. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize which fault is reported.</p> <p>For all other exceptions taken to EL1, FAR_EL1 is UNKNOWN.</p> <p>FAR_EL1 is made UNKNOWN on an exception return from EL1.</p>	64 {x}

Access

MRS <Xt>, FAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b000

MSR FAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0000	0b000

Accessibility

MRS <Xt>, FAR_EL1

```

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

```
    else
        X[t, 64] = FAR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = FAR_EL1;
```

MSR FAR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        FAR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    FAR_EL1 = X[t, 64];
```

A.2.2.28 PRENR_EL1, Protection Region Enable Register (EL1)

Provides direct access to the AArch64-PRLAR_EL1.EN bits of EL1 MPU regions from 0 to 31.

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	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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-76: AARCH64_PRENR_EL1 bit assignments

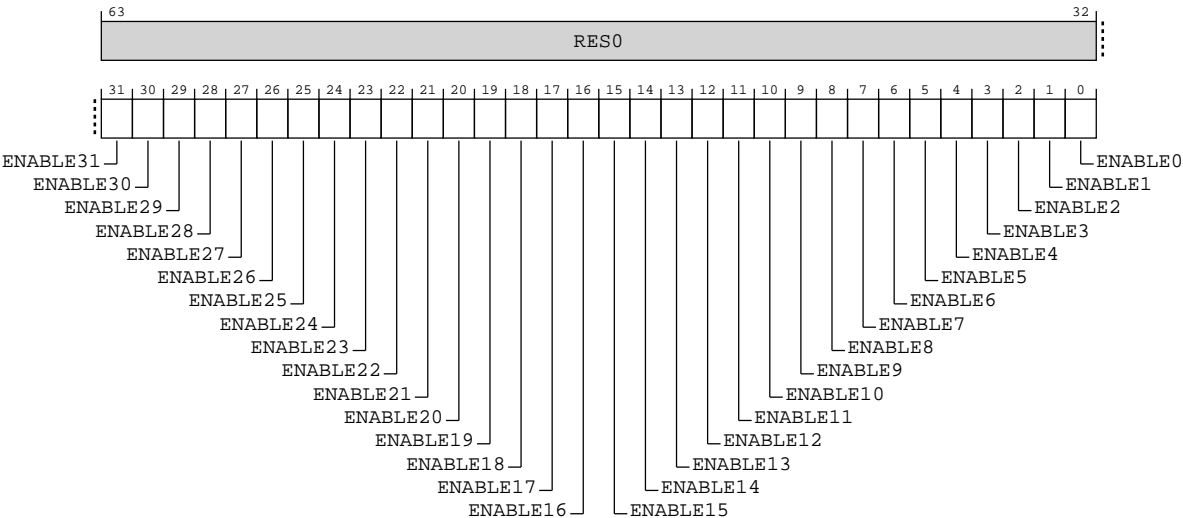


Table A-192: PRENR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ENABLE<n>, bit[n], where n = 31 to 0	Enable bit. Each bit, n, enables or disables the respective EL1 MPU region. The bits associated with the unimplemented MPU regions are RAZ/WI . 0b0 Disables the EL1 MPU n region. 0b1 Enables the EL1 MPU n region.	0x00000000

Access

MRS <Xt>, PRENR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0001	0b001

MSR PRENR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0001	0b001

Accessibility

MRS <Xt>, PRENR_EL1

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

```
if HCR_EL2.TVM == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif VTCR_EL2.MSA == '1' then
    UNDEFINED;
else
    X[t, 64] = PRENR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PRENR_EL1;
```

MSR PRENR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        PRENR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    PRENR_EL1 = X[t, 64];
```

A.2.2.29 PRSELR_EL1, Protection Region Selection Register (EL1)

Selects the region number for the EL1 MPU region associated with the AArch64-PRBAR_EL1 and AArch64-PRLAR_EL1 registers.

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-77: AARCH64_PRSELR_EL1 bit assignments

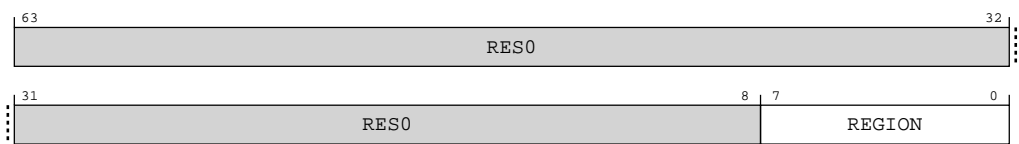


Table A-195: PRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	REGION	The number of the current EL1 MPU region visible in AArch64-PRBAR_EL1 and AArch64-PRLAR_EL1. For N implemented MPU regions, memory region numbering starts at 0 and increments by 1 to the value N-1. Writing a value greater than or equal to the number of implemented MPU regions specified by AArch64-MPUIR_EL1.REGION, results in CONSTRAINED UNPREDICTABLE behavior. CONSTRAINED UNPREDICTABLE behavior is that PRSELR_EL1 register becomes UNKNOWN .	8{x}

Access

MRS <Xt>, PRSELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0010	0b001

MSR PRSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b0010	0b001

Accessibility

MRS <Xt>, PRSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        X[t, 64] = PRSELR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRSELR_EL1;
```

MSR PRSELR_EL1, <Xt>

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



```
elseif PSTATE.EL == EL1 then
  if HCR_EL2.TVM == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  elseif VTCR_EL2.MSA == '1' then
    UNDEFINED;
  else
    PRSELR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
  PRSELR_EL1 = X[t, 64];
```

A.2.2.30 PRBAR_EL1, Protection Region Base Address Register (EL1)

Provides access to the base addresses for the EL1 MPU region. AArch64-PRSELR_EL1.REGION determines which MPU region is selected.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-78: AARCH64_PRBAR_EL1 bit assignments

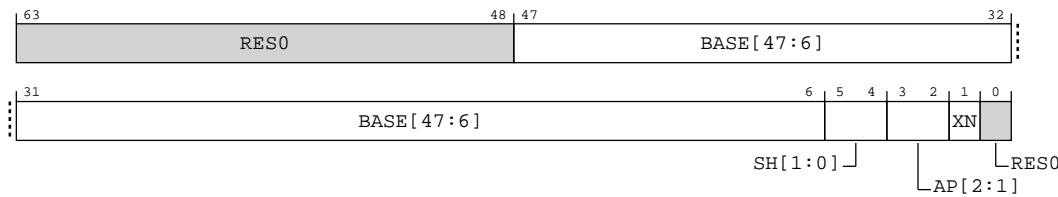


Table A-198: PRBAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:6]	BASE[47:6]	Bits[47:6] of the lower inclusive limit of the selected EL1 MPU memory region. This value is zero extended to provide the base address to be checked against.	42 {x}
[5:4]	SH[1:0]	Shareability attribute. 0b00 Non-shareable 0b01 Reserved, CONSTRAINED UNPREDICTABLE 0b10 Outer Shareable 0b11 Inner Shareable	xx
[3:2]	AP[2:1]	Access Permission attributes. 0b00 Read/write at EL1, no access at EL0 0b01 Read/write at EL1 and EL0 0b10 Read-only at EL1, no access at EL0 0b11 Read-only at EL1 and EL0	xx
[1]	XN	Execute Never 0b0 Execution of instructions fetched from the region is permitted. 0b1 Execution of instructions fetched from the region is not permitted.	x
[0]	RES0	Reserved	RES0

Access

Any access to MPU region register PRBAR_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR_EL1 register return an **UNKNOWN** value.
- Writes to unimplemented PRBAR_EL1 register make all PRBAR_EL1 registers value **UNKNOWN**.

MRS <Xt>, PRBAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b1000	0b000

MSR PRBAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b1000	0b000

Accessibility

Any access to MPU region register PRBAR_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is CONSTRAINED UNPREDICTABLE.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR_EL1 register return an UNKNOWN value.
- Writes to unimplemented PRBAR_EL1 register make all PRBAR_EL1 registers value UNKNOWN.

MRS <Xt>, PRBAR_EL1

```
if UInt(PRSEL_EL1.REGION) >= UInt(MPUIR_EL1.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        X[t, 64] = PRBAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)];
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRBAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)];
```

MSR PRBAR_EL1, <Xt>

```
if UInt(PRSEL_EL1.REGION) >= UInt(MPUIR_EL1.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        PRBAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)] = X[t, 64];
elseif PSTATE.EL == EL2 then
    PRBAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)] = X[t, 64];
```

A.2.2.31 PRLAR_EL1, Protection Region Limit Address Register (EL1)

Provides access to the limit addresses for the EL1 MPU region. AArch64-PRSEL_EL1.REGION determines which MPU region is selected.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-79: AARCH64_PRLAR_EL1 bit assignments

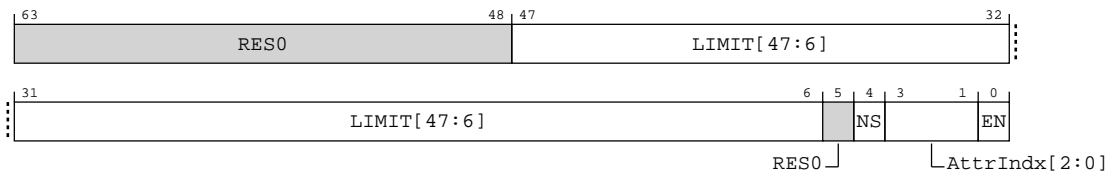


Table A-201: PRLAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:6]	LIMIT[47:6]	Bits[47:6] of the upper inclusive limit of the selected EL1 MPU memory region. This value is concatenated with the value 0x3F to provide the limit address to be checked against.	42 {x}
[5]	RES0	Reserved	RES0
[4]	NS	Non-secure bit. Specifies whether the output address is in the Secure or Non-secure memory. 0b0 Output address is in Secure address space. 0b1 Output address is in Non-secure address space.	x

Bits	Name	Description	Reset
[3:1]	AttrIdx[2:0]	<p>Selects attributes from within the associated Memory Attribute Indirection Register.</p> <p>0b000 Select the Attr0 field from MAIR_EL1.</p> <p>0b001 Select the Attr1 field from MAIR_EL1.</p> <p>0b010 Select the Attr2 field from MAIR_EL1.</p> <p>0b011 Select the Attr3 field from MAIR_EL1.</p> <p>0b100 Select the Attr4 field from MAIR_EL1.</p> <p>0b101 Select the Attr5 field from MAIR_EL1.</p> <p>0b110 Select the Attr6 field from MAIR_EL1.</p> <p>0b111 Select the Attr7 field from MAIR_EL1.</p>	xxx
[0]	EN	<p>Region enable bit.</p> <p>0b0 Region disabled.</p> <p>0b1 Region enabled.</p>	0b0

Access

Any access to MPU region register PRLAR_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR_EL1 register return an **UNKNOWN** value.
- Writes to unimplemented PRLAR_EL1 register make all PRLAR_EL1 registers value **UNKNOWN**.

MRS <Xt>, PRLAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b1000	0b001

MSR PRLAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0110	0b1000	0b001

Accessibility

Any access to MPU region register PRLAR_EL1 above the number of implemented regions specified by AArch64-MPUIR_EL1.REGION is CONSTRAINED UNPREDICTABLE.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR_EL1 register return an UNKNOWN value.
- Writes to unimplemented PRLAR_EL1 register make all PRLAR_EL1 registers value UNKNOWN.

MRS <Xt>, PRLAR_EL1

```

if UInt(PRSEL_EL1.REGION) >= UInt(MPUIR_EL1.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        X[t, 64] = PRLAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)];
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRLAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)];

```

MSR PRLAR_EL1, <Xt>

```

if UInt(PRSEL_EL1.REGION) >= UInt(MPUIR_EL1.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif VTCR_EL2.MSA == '1' then
        UNDEFINED;
    else
        PRLAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)] = X[t, 64];
elseif PSTATE.EL == EL2 then
    PRLAR_EL1[UInt(AArch64-PRSEL_EL1.REGION)] = X[t, 64];

```

A.2.2.32 PAR_EL1, Physical Address Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group
Generic System Control

Access type
See bit descriptions

Reset value

When AArch64-PAR_EL1.F == '0'

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

When AArch64-PAR_EL1.F == '1'

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

When AArch64-PAR_EL1.F == '0'

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

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

- The PAR_EL1.{ATTR, SH} fields are permitted to report the resulting attributes, instead of reporting the values that appear in the translation table descriptors.
- See the PAR_EL1.NS bit description for constraints on the value it returns.

Figure A-80: AARCH64_PAR_EL1 bit assignments

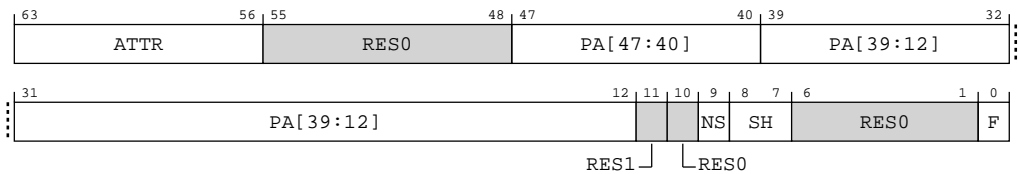


Table A-204: PAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	ATTR	Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in AArch64-MAIR_EL1 and AArch64-MAIR_EL2. The value returned in this field can be the resulting attribute, instead of the value that appears in the translation table descriptor.	8 {x}

Bits	Name	Description	Reset
[55:48]	RES0	Reserved	RES0
[47:40]	PA[47:40]	<p>When PA_W == 48</p> <p>Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[47:40].</p> <p>Otherwise</p> <p>RES0</p>	8 {x}
[39:12]	PA[39:12]	Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[39:12].	28 {x}
[11]	RES1	Reserved	RES1
[10]	RES0	Reserved	RES0
[9]	NS	<p>Non-secure. The NS attribute for a translation table entry from a Secure translation regime.</p> <p>This bit reflects the Security state of the intermediate physical address space of the translation for the instructions:</p> <ul style="list-style-type: none"> AT S1E1R, AT S1E1W, AT S1E1RP, AT S1E1WP, AT S1E0R, and AT S1E0W. <p>Otherwise, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.</p>	x
[8:7]	SH	<p>Shareability attribute, for the returned output address. Permitted values are:</p> <p>0b00</p> <p>Non-shareable.</p> <p>0b10</p> <p>Outer Shareable.</p> <p>0b11</p> <p>Inner Shareable.</p> <p>The value 0b01 is reserved.</p> <p>Note:</p> <p>This field returns the value 0b10 for:</p> <ul style="list-style-type: none"> Any type of Device memory. Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes. <p>The value returned in this field can be the resulting attribute, instead of the value that appears in the translation table descriptor.</p>	xx
[6:1]	RES0	Reserved	RES0
[0]	F	<p>Indicates whether the instruction performed a successful address translation.</p> <p>0b0</p> <p>Address translation completed successfully.</p>	x

When AArch64-PAR_EL1.F == '1'

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

Figure A-81: AARCH64_PAR_EL1 bit assignments

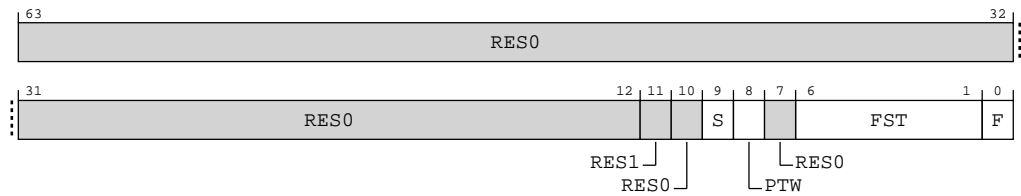


Table A-205: PAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	RES0
[11]	RES1	Reserved	RES1
[10]	RES0	Reserved	RES0
[9]	S	Indicates the translation stage at which the translation aborted: 0b0 Translation aborted because of a fault in the stage 1 translation. 0b1 Translation aborted because of a fault in the stage 2 translation.	x
[8]	PTW	If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.	x
[7]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[6:1]	FST	<p>Fault status code, as shown in the Data Abort ESR encoding.</p> <p>0b000000 Address size fault, level 0 of translation or translation table base register.</p> <p>0b000001 Address size fault, level 1.</p> <p>0b000010 Address size fault, level 2.</p> <p>0b000011 Address size fault, level 3.</p> <p>0b000100 Translation fault, level 0.</p> <p>0b000101 Translation fault, level 1.</p> <p>0b000110 Translation fault, level 2.</p> <p>0b000111 Translation fault, level 3.</p> <p>0b001001 Access flag fault, level 1.</p> <p>0b001010 Access flag fault, level 2.</p> <p>0b001011 Access flag fault, level 3.</p> <p>0b001100 Permission fault, level 0.</p> <p>0b001101 Permission fault, level 1.</p> <p>0b001110 Permission fault, level 2.</p> <p>0b001111 Permission fault, level 3.</p> <p>0b010100 Synchronous External abort on translation table walk or hardware update of translation table, level 0.</p> <p>0b010101 Synchronous External abort on translation table walk or hardware update of translation table, level 1.</p> <p>0b010110 Synchronous External abort on translation table walk or hardware update of translation table, level 2.</p> <p>0b010111 Synchronous External abort on translation table walk or hardware update of translation table, level 3.</p> <p>0b110000 TLB conflict abort.</p> <p>0b110001 Unsupported atomic hardware update fault.</p>	6 {x}

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

Access

MRS <Xt>, PAR_EL1

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

MSR PAR_EL1, <Xt>

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

Accessibility

MRS <Xt>, PAR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    X[t, 64] = PAR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PAR_EL1;

```

MSR PAR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    PAR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    PAR_EL1 = X[t, 64];

```

A.2.2.33 MAIR_EL1, Memory Attribute Indirection Register (EL1)

If VM8Av8-64 is enabled at stage 1 of EL1&0 translation regime, this register provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations at EL1.

If PM8Av8-64 is enabled at stage 1 of EL1&0 translation regime, this register provides the memory attribute encodings corresponding to the possible AttrIdx values in AArch64-PRLAR_EL1 register for stage 1 translations.

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

MAIR_EL1 is permitted to be cached in a TLB.

Figure A-82: AARCH64_MAIR_EL1 bit assignments

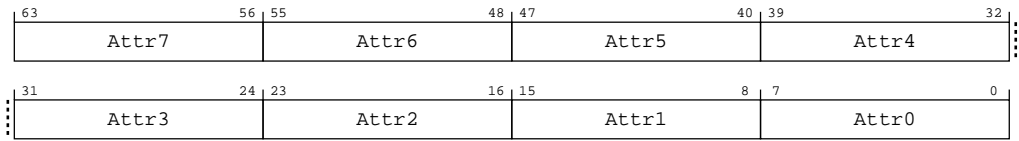


Table A-208: MAIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Attr<n>	<p>The memory attribute encoding for an AttrIdx[2:0] gives the value of <n> in Attr<n>.</p> <p>Attr is encoded as follows: Table A-209: Attr description on page 489</p> <p>'dd' is encoded as follows: Table A-210: dd description on page 489</p> <p>'oooo' is encoded as follows: Table A-211: 'oooo' description on page 489</p> <p>R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.</p> <p>'iiii' is encoded as follows: Table A-212: 'iiii' description on page 489</p> <p>R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.</p> <p>The R and W bits in 'oooo' and 'iiii' fields have the following meanings: Table A-213: R or W description on page 489</p>	64 {x}

Table A-209: Attr description

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000ddxx, (xx != 00)	UNPREDICTABLE.
0booooiiii, (oooo != 0000 and iiii != 0000)	Normal memory. See encoding of 'oooo' and 'iiii' for the type of Normal Memory.
0bxxxx0000, where xxxx != 0000	UNPREDICTABLE.

Table A-210: dd description

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

Table A-211: 'oooo' description

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

Table A-212: 'iiii' description

'iiii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

Table A-213: R or W description

R or W	Meaning
0b0	No Allocate
0b1	Allocate

Access

MRS <Xt>, MAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000

MSR MAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000

Accessibility

MRS <Xt>, MAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MAIR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MAIR_EL1;
```

MSR MAIR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        MAIR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    MAIR_EL1 = X[t, 64];
```

A.2.2.34 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

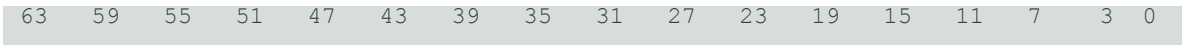
Generic System Control

Access type

See bit descriptions

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-83: AARCH64_AMAIR_EL1 bit assignments

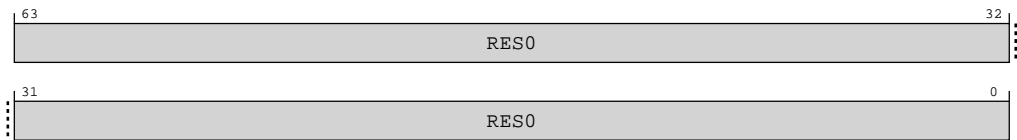


Table A-216: AMAIR_EL1 bit descriptions

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

Access

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

MRS <Xt>, AMAIR_EL1

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

MSR AMAIR_EL1, <Xt>

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

```
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AMAIR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    AMAIR_EL1 = X[t, 64];
```

A.2.2.35 VBAR_EL1, Vector Base Address Register (EL1)

Holds the vector base address for any exception that is 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-84: AARCH64_VBAR_EL1 bit assignments

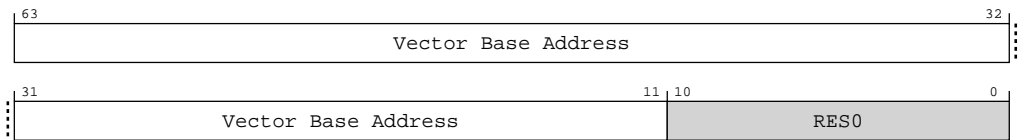


Table A-219: VBAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:11]	None	Vector Base Address. Base address of the exception vectors for exceptions taken to EL1. Note: <ul style="list-style-type: none"> If tagged addresses are being used, bits [55:48] of VBAR_EL1 must be the same or else the use of the vector address will result in a recursive exception. If tagged addresses are not being used, bits [63:48] of VBAR_EL1 must be the same or else the use of the vector address will result in a recursive exception. 	53 {x}
[10:0]	RES0	Reserved	RES0

Access

MRS <Xt>, VBAR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

MSR VBAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

Accessibility

MRS <Xt>, VBAR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    X[t, 64] = VBAR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = VBAR_EL1;

```

MSR VBAR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    VBAR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    VBAR_EL1 = X[t, 64];

```

A.2.2.36 ISR_EL1, Interrupt Status Register

Shows the pending status of the IRQ, FIQ, or SError interrupt.

When executing at EL2, this shows the pending status of the physical IRQ, FIQ, or SError interrupts.

When executing at EL1:

- If the AArch64-HCR_EL2.{IMO,FMO,AMO} bit has a value of 1, the corresponding ISR_EL1.{I,F,A} bit shows the pending status of the virtual IRQ, FIQ, or SError.
- If the AArch64-HCR_EL2.{IMO,FMO,AMO} bit has a value of 0, the corresponding ISR_EL1.{I,F,A} bit shows the pending status of the physical IRQ, FIQ, or SError.

Configurations

This register is available in all configurations.

Attributes

Width

64

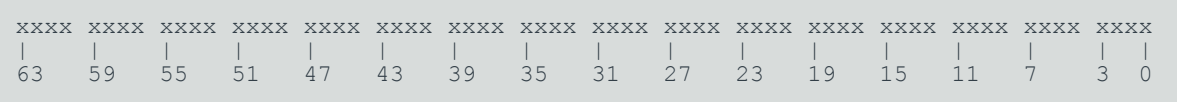
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-85: AARCH64_ISR_EL1 bit assignments

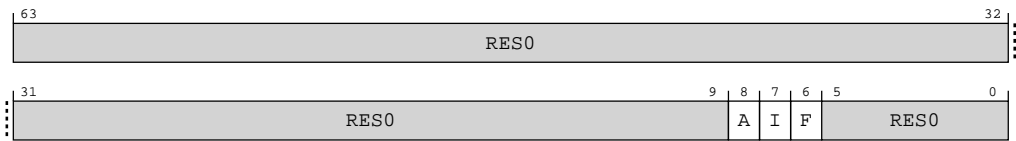


Table A-222: ISR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:9]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[8]	A	SErrror interrupt pending bit. Indicates whether an SErrror interrupt is pending. 0b0 No pending SErrror. 0b1 An SErrror interrupt is pending. If the SErrror interrupt is edge-triggered, this field is cleared to zero when the physical SErrror interrupt is taken.	x
[7]	I	IRQ pending bit. Indicates whether an IRQ interrupt is pending. 0b0 No pending IRQ. 0b1 An IRQ interrupt is pending. Note: This bit indicates the presence of a pending IRQ interrupt regardless of whether the interrupt has Superpriority.	x
[6]	F	FIQ pending bit. Indicates whether an FIQ interrupt is pending. 0b0 No pending FIQ. 0b1 An FIQ interrupt is pending. Note: This bit indicates the presence of a pending FIQ interrupt regardless of whether the interrupt has Superpriority.	x
[5:0]	RES0	Reserved	RES0

Access

MRS <Xt>, ISR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0001	0b000

Accessibility

MRS <Xt>, ISR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = ISR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ISR_EL1;

```

A.2.2.37 CONTEXTIDR_EL1, Context ID Register (EL1)

Identifies the current Process Identifier.

The value of the whole of this register is called the Context ID and is used by:

- The debug logic, for Linked and Unlinked Context ID matching.
- The trace logic, to identify the current process.

The significance of this register is for debug and trace use only.

Configurations

This register is available in all configurations.

Attributes

Width

64

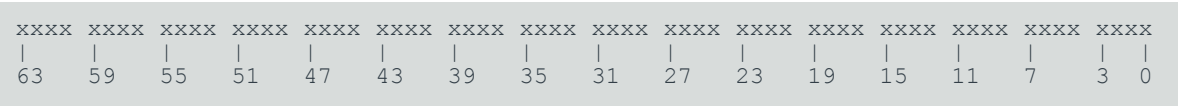
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-86: AARCH64_CONTEXTIDR_EL1 bit assignments

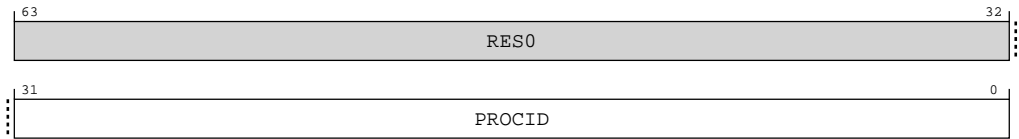


Table A-224: CONTEXTIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:0]	PROCID	Process Identifier. This field must be programmed with a unique value that identifies the current process. Note: In AArch64 state, CONTEXTIDR_EL1 is independent of the ASID, and for the EL1&0 translation regime either AArch64-TTBRO_EL1 or AArch64-TTBR1_EL1 holds the ASID.	32 {x}

Access

MRS <Xt>, CONTEXTIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b001

MSR CONTEXTIDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b001

Accessibility

MRS <Xt>, CONTEXTIDR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = CONTEXTIDR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CONTEXTIDR_EL1;

```

MSR CONTEXTIDR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        CONTEXTIDR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    CONTEXTIDR_EL1 = X[t, 64];

```

A.2.2.38 TPIDR_EL1, EL1 Software Thread ID Register

Provides a location where software executing at EL1 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

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-87: AARCH64_TPIDR_EL1 bit assignments

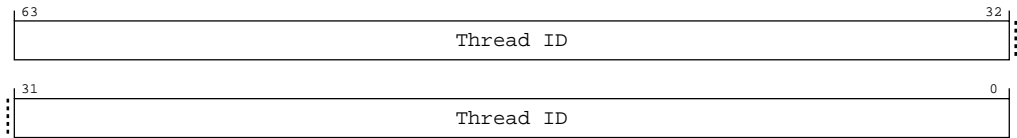


Table A-227: TPIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Thread ID. Thread identifying information stored by software running at this Exception level.	64 {x}

Access

MRS <Xt>, TPIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b100

MSR TPIDR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1101	0b0000	0b100

Accessibility

MRS <Xt>, TPIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = TPIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = TPIDR_EL1;
```

MSR TPIDR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    TPIDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    TPIDR_EL1 = X[t, 64];
```

A.2.2.39 IMP_ITCMREGIONR_EL1, ITCM Region Register

This register controls the ITCM region.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

inconsistent

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-88: AARCH64_IMP_ITCMREGIONR_EL1 bit assignments

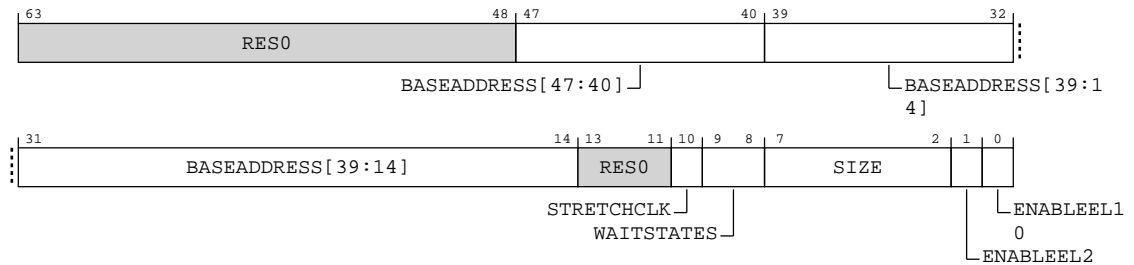


Table A-230: IMP_ITCMREGIONR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	BASEADDRESS[47:40]	When PA_W == 48 TCM base address, bits [47:40]. This field is read-only. This field takes the value of the CFGITCMBASEADDRm[47:40] configuration pins. Otherwise RES0	8 {x}
[39:14]	BASEADDRESS[39:14]	TCM base address, bits [39:14] (aligned to TCM size). This field is read-only. This field takes the value of the CFGITCMBASEADDRm[39:14] configuration pins. Access to this field is: RO	26 {x}
[13:11]	RES0	Reserved	RES0
[10]	STRETCHCLK	TCM clock stretch. This field is read-only. This field takes the value of the ITCM_STRETCH_CLKm core parameter. 0b0 No clock stretch. 0b1 Clock stretched to occupy full cycle. Access to this field is: RO	x
[9:8]	WAITSTATES	TCM accesses wait states: 0-3 cycles. This field is read-only. This field takes the value of the ITCM_WAITm core parameter. Access to this field is: RO	xx

Bits	Name	Description	Reset
[7:2]	SIZE	<p>TCM size, encoded as $1 + \log_2(\text{size}/1\text{KB})$. This field is read-only. This field takes the value of the ITCM_SIZE_m core parameter.</p> <p>0b000000 no TCM present.</p> <p>0b000101 16 KB.</p> <p>0b000110 32 KB.</p> <p>0b000111 64 KB.</p> <p>0b001000 128 KB.</p> <p>0b001001 256 KB.</p> <p>0b001010 512 KB.</p> <p>0b001011 1 MB.</p> <p>Access to this field is: RO</p>	6 {x}
[1]	ENABLEEL2	<p>TCM enable at EL2. This field is read-only in EL1 or EL0. This field resets to the value of the CFGITCMEN_m configuration pin.</p> <p>0b0 TCM disabled at EL2.</p> <p>0b1 TCM enabled at EL2.</p> <p>When PSTATE.EL == EL2 Access to this field is: RW</p> <p>Otherwise Access to this field is: RO</p>	x
[0]	ENABLEEL10	<p>TCM enable at EL1 and EL0. If the stage 1 EL1&0 translation regime uses the VMSAv8-64 memory architecture, this field reads and behaves as 0 and ignores writes.</p> <p>0b0 TCM disabled at EL1 and EL0.</p> <p>0b1 TCM enabled at EL1 and EL0.</p> <p>When PSTATE.EL == EL2 or AArch64-VTCR_EL2.MSA == '0' Access to this field is: RW</p> <p>Otherwise Access to this field is: RAZ/WI</p>	0b0

Access

MRS <Xt>, IMP_ITCMREGIONR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b11111	0b0000	0b001

MSR IMP_ITCMREGIONR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b11111	0b0000	0b001

Accessibility

MRS <Xt>, IMP_ITCMREGIONR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_ITCMREGIONR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_ITCMREGIONR_EL1;
```

MSR IMP_ITCMREGIONR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.REGIONS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ITCMREGIONR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_ITCMREGIONR_EL1 = X[t, 64];
```

A.2.2.40 IMP_DTCMREGIONR_EL1, DTCM Region Register

This register controls the DTCM region.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

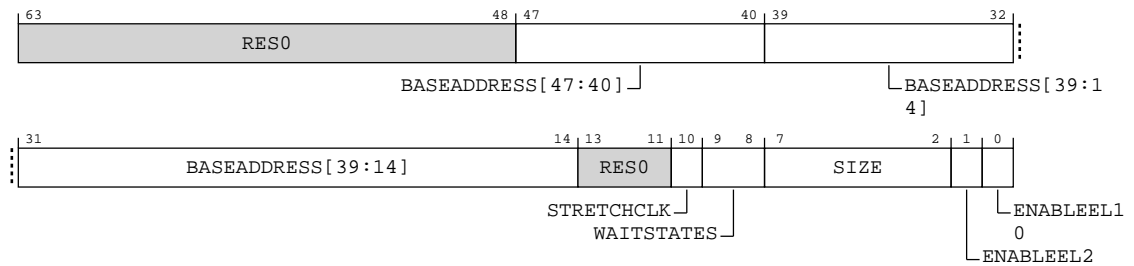
inconsistent

Reset value

xxxx	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-89: AARCH64_IMP_DTCMREGIONR_EL1 bit assignments****Table A-233: IMP_DTCMREGIONR_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	BASEADDRESS[47:40]	When PA_W == 48 TCM base address, bits [47:40]. This field is read-only. This field takes the value of the CFGDTCMBASEADDRm[47:40] configuration pins. Otherwise RES0	8 {x}
[39:14]	BASEADDRESS[39:14]	TCM base address, bits [39:14] (aligned to TCM size). This field is read-only. This field takes the value of the CFGDTCMBASEADDRm[39:14] configuration pins. Access to this field is: RO	26 {x}
[13:11]	RES0	Reserved	RES0
[10]	STRETCHCLK	TCM clock stretch. This field is read-only. This field takes the value of the DTCM_STRETCH_CLKm core parameter. 0b0 No clock stretch. 0b1 Clock stretched to occupy full cycle. Access to this field is: RO	x

Bits	Name	Description	Reset
[9:8]	WAITSTATES	TCM accesses wait states: 0-3 cycles. This field is read-only. This field takes the value of the DTCM_WAITm core parameter. Access to this field is: RO	xx
[7:2]	SIZE	TCM size, encoded as 1+log2(size/1KB). This field is read-only. This field takes the value of the DTCM_SIZEEm core parameter. 0b000000 no TCM present. 0b000101 16 KB. 0b000110 32 KB. 0b000111 64 KB. 0b001000 128 KB. 0b001001 256 KB. 0b001010 512 KB. 0b001011 1 MB. Access to this field is: RO	6{x}
[1]	ENABLEEL2	TCM enable at EL2. This field is read-only in EL1 or EL0. 0b0 TCM disabled at EL2. 0b1 TCM enabled at EL2. When PSTATE.EL == EL2 Access to this field is: RW Otherwise Access to this field is: RO	0b0
[0]	ENABLEEL10	TCM enable at EL1 and EL0. If the stage 1 EL1&0 translation regime uses the VMSAv8-64 memory architecture, this field reads and behaves as 0 and ignores writes. 0b0 TCM disabled at EL1 and EL0. 0b1 TCM enabled at EL1 and EL0. When PSTATE.EL == EL2 or AArch64-VTCR_EL2.MSA == '0' Access to this field is: RW Otherwise Access to this field is: RAZ/WI	0b0

Access

MRS <Xt>, IMP_DTCMREGIONR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b010

MSR IMP_DTCMREGIONR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b010

Accessibility

MRS <Xt>, IMP_DTCMREGIONR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_DTCMREGIONR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_DTCMREGIONR_EL1;
```

MSR IMP_DTCMREGIONR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.REGIONS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_DTCMREGIONR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_DTCMREGIONR_EL1 = X[t, 64];
```

A.2.2.41 IMP_LLPPREGIONR_EL1, LLPP Region Register

This register controls the LLPP region.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

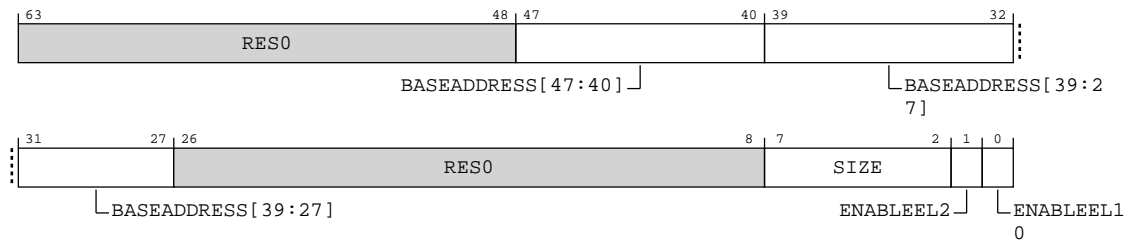
RO

Reset value

xxxx	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



Where the reset reads xxxx, see individual bits.

Bit descriptions**Figure A-90: AARCH64_IMP_LLPPREGIONR_EL1 bit assignments****Table A-236: IMP_LLPPREGIONR_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	BASEADDRESS[47:40]	When PA_W == 48 LLPP base address, bits [47:40]. This field is read-only. This field takes the value of the CFGLLPPBASE[47:40] configuration pins. Otherwise RES0	8 {x}
[39:27]	BASEADDRESS[39:27]	LLPP base address, bits [39:27] (aligned to 128 MB). This field is read-only. This field takes the value of the CFGLLPPBASE[39:27] configuration pins. Access to this field is: RO	13 {x}
[26:8]	RES0	Reserved	RES0
[7:2]	SIZE	LLPP size, encoded as $1 + \log_2(\text{size}/1\text{KB})$. This field is read-only. 0b000000 no LLPP present (when the CFGLLPPIMP configuration pin is 0x0) 0b010010 128 MB (when the CFGLLPPIMP configuration pin is 0x1) Access to this field is: RO	6 {x}

Bits	Name	Description	Reset
[1]	ENABLEEL2	<p>LLPP enable at EL2. This field is read-only in EL1 or EL0.</p> <p>0b0</p> <p>LLPP disabled at EL2.</p> <p>0b1</p> <p>LLPP enabled at EL2.</p> <p>When PSTATE.EL == EL2</p> <p>Access to this field is: RW</p> <p>Otherwise</p> <p>Access to this field is: RO</p>	0b0
[0]	ENABLEEL10	<p>LLPP enable at EL1 and EL0.</p> <p>0b0</p> <p>LLPP disabled at EL1 and EL0.</p> <p>0b1</p> <p>LLPP enabled at EL1 and EL0.</p>	0b0

Access

MRS <Xt>, IMP_LLPPREGIONR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b011

MSR IMP_LLPPREGIONR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b011

Accessibility

MRS <Xt>, IMP_LLPPREGIONR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_LLPPREGIONR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_LLPPREGIONR_EL1;

```

MSR IMP_LLPPREGIONR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.REGIONS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_LLPPREGIONR_EL1 = X[t, 64];

```

```
elseif PSTATE.EL == EL2 then
    IMP_LLPPREGIONR_EL1 = X[t, 64];
```

A.2.2.42 IMP_LLRAMREGIONR_EL1, LLRAM Region Register

This register controls the LLRAM region.

Configurations

This register is available in all configurations.

Attributes

Width

64

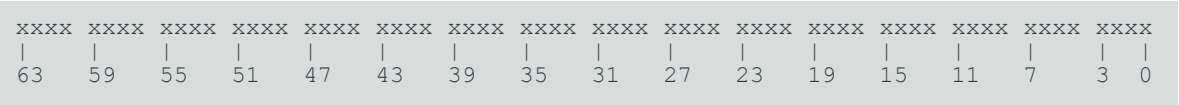
Functional group

Generic System Control

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-91: AARCH64_IMP_LLRAMREGIONR_EL1 bit assignments

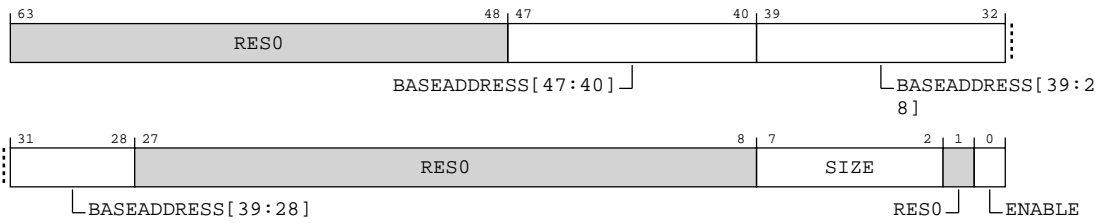


Table A-239: IMP_LLRAMREGIONR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[47:40]	BASEADDRESS[47:40]	When PA_W == 48 LLRAM base address, bits [47:40]. This field is read-only. This field takes the value of the CFGLLRAMBASE[47:40] configuration pins. Otherwise RES0	8 {x}
[39:28]	BASEADDRESS[39:28]	LLRAM base address, bits [39:28] (aligned to 256 MB). This field is read-only. This field takes the value of the CFGLLRAMBASE[39:28] configuration pins. Access to this field is: RO	12 {x}
[27:8]	RES0	Reserved	RES0
[7:2]	SIZE	LLRAM size, encoded as $1 + \log_2(\text{size}/1\text{KB})$. This field is read-only. 0b000000 no LLRAM present (when the CFGLLRAMIMP configuration pin is 0x0) 0b010011 256 MB (when the CFGLLRAMIMP configuration pin is 0x1) Access to this field is: RO	6 {x}
[1]	RES0	Reserved	RES0
[0]	ENABLE	LLRAM enable. This field resets to the value of the CFGLLRAMEN configuration pin. Note: This bit controls whether the current core has access to the LLRAM region. It does not enable or disable the LLRAM itself. The LLRAM can be accessed by other cores and can service LLRAM ACP requests regardless of this bit being 0. 0b0 LLRAM region disabled. 0b1 LLRAM region enabled.	x

Access

MRS <Xt>, IMP_LLRAMREGIONR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b100

MSR IMP_LLRAMREGIONR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b100

Accessibility

MRS <Xt>, IMP_LLRAMREGIONR_EL1

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

```
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_LLRAMREGIONR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_LLRAMREGIONR_EL1;
```

MSR IMP_LLRAMREGIONR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.REGIONS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_LLRAMREGIONR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_LLRAMREGIONR_EL1 = X[t, 64];
```

A.2.2.43 IMP_SPPREGIONR_EL1, SPP Region Register

This register controls the SPP region.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RO

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-92: AARCH64_IMP_SPPREGIONR_EL1 bit assignments

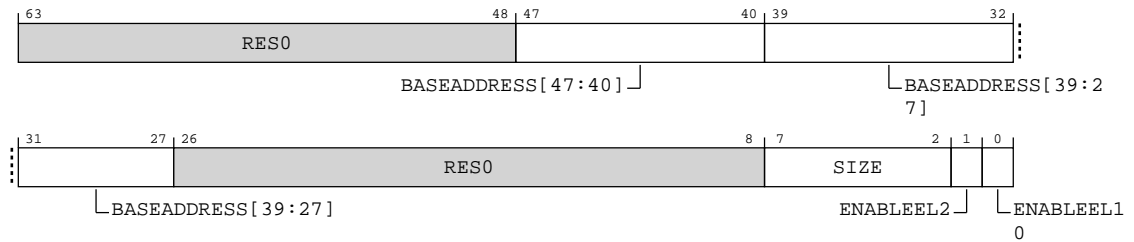


Table A-242: IMP_SPPREGIONR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	BASEADDRESS[47:40]	When PA_W == 48 SPP base address, bits [47:40]. This field is read-only. This field takes the value of the CFGSPPBASE[47:40] configuration pins. Otherwise RES0	8 {x}
[39:27]	BASEADDRESS[39:27]	SPP base address, bits [39:27] (aligned to 128 MB). This field is read-only. This field takes the value of the CFGSPPBASE[39:27] configuration pins. Access to this field is: RO	13 {x}
[26:8]	RES0	Reserved	RES0
[7:2]	SIZE	SPP size, encoded as $1 + \log_2(\text{size}/1\text{KB})$. This field is read-only. 0b000000 No SPP present (when the CFGSPPIMP configuration pin is 0x0) 0b010010 128 MB (when the CFGSPPIMP configuration pin is 0x1) Access to this field is: RO	6 {x}
[1]	ENABLEEL2	SPP enable at EL2. This field is read-only in EL1 or EL0. Note: This bit controls whether the current core has access to the SPP region from EL2. The SPP can be accessed by other cores from EL2, regardless of this bit being 0. 0b0 SPP disabled at EL2. 0b1 SPP enabled at EL2. When PSTATE.EL == EL2 Access to this field is: RW Otherwise Access to this field is: RO	0b0

Bits	Name	Description	Reset
[0]	ENABLEEL10	<p>SPP enable at EL1 and EL0.</p> <p>Note: This bit controls whether the current core has access to the SPP region from EL1 and EL0. The SPP can be accessed by other cores from EL1 and EL0, regardless of this bit being 0.</p> <p>0b0 SPP disabled at EL1 and EL0.</p> <p>0b1 SPP enabled at EL1 and EL0.</p>	0b0

Access

MRS <Xt>, IMP_SPPREGIONR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b101

MSR IMP_SPPREGIONR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0000	0b101

Accessibility

MRS <Xt>, IMP_SPPREGIONR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_SPPREGIONR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_SPPREGIONR_EL1;

```

MSR IMP_SPPREGIONR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.REGIONS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_SPPREGIONR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_SPPREGIONR_EL1 = X[t, 64];

```

A.2.2.44 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register

This register contains control bits that affect the CPU behavior.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	x001	0001	0100	1110	1101	0xx0	1111	0011	1000	0111	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-93: AARCH64_IMP_CPUACTLR_EL1 bit assignments

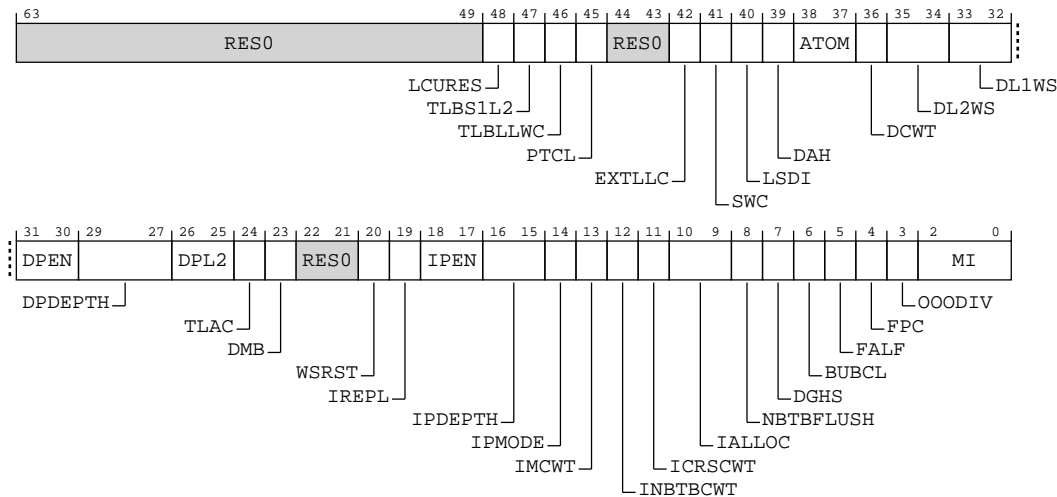


Table A-245: IMP_CPUACTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:49]	RES0	Reserved	RES0
[48]	LCURES	<p>When CFGLLRAMIMP == 1 or CFGSPPIMP == 1</p> <p>Reserve some CPU resources for LCU (LLRAM and SPP) accesses.</p> <p>0b0</p> <p>No specific resources are reserved for LLRAM or SPP accesses.</p> <p>0b1</p> <p>A certain number of STB slots and I-side and D-side linefill descriptors are reserved to ensure that LLRAM and SPP accesses do not have increased worst case latency caused by Main Manager accesses.</p> <p>When this bit is set to 0, accesses to Main Manager can fill up some resources in the CPU (STB slots and linefill descriptors) and the latency of subsequent LLRAM and SPP accesses may be impacted until older transactions have completed and freed these resources.</p> <p>If the value of AArch64-IMP_INTLATENCY_EL2.LCURES is not 0 then PE behaves as if the value of the AArch64-IMP_CPUACTLR_EL1.LCURES field is 1 for all purposes other than returning the value of a direct read of the field.</p> <p>Otherwise</p> <p>RES0</p>	'0'
[47]	TLBS1L2	<p>When VMSAm == 1</p> <p>TLB S1L2 walk caching behavior.</p> <p>0b0</p> <p>TLB S1L2 walk caching disabled.</p> <p>0b1</p> <p>TLB S1L2 walk caching enabled.</p> <p>Otherwise</p> <p>RES0</p>	'1'
[46]	TLBLLWC	<p>When VMSAm == 1</p> <p>TLB Last Level Walk Caching behavior.</p> <p>0b0</p> <p>TLB last level walk caching disabled.</p> <p>0b1</p> <p>TLB last level walk caching enabled.</p> <p>Otherwise</p> <p>RES0</p>	'1'

Bits	Name	Description	Reset
[45]	PTCL	When VMSAm == 1 Page Tables Caching Level. 0b0 Cache page tables in L1. 0b1 Cache page tables in L2. Otherwise RES0	'0'
[44:43]	RES0	Reserved	RES0
[42]	EXTLLC	External Last-Level Cache present. Affects the counting of cache-related PMU events. Used to control how the LL_CACHE* PMU events count. 0b0 The last-level cache is the processor L2. 0b1 An external Last-level cache is present in the system. If the MM uses CHI, the DataSource field will indicate when data is returned from the external cache.	0b0
[41]	SWC	Store buffer Write streaming Cache lookup behavior. 0b0 Store buffer skips cache lookups while in write-streaming mode. 0b1 Store buffer performs cache lookups while in write-streaming mode.	0b0
[40]	LSDI	Load/store dual-issuing control. 0b0 Dual-issuing of load/store instructions disabled. Only the first memory pipeline will be used. Note that a load or store can still be issued with a non-memory instruction. 0b1 Dual-issuing of load/store instructions enabled.	0b1
[39]	DAH	Disable Allocation Hints. 0b0 Normal operation. 0b1 The Inner Allocation and Transient Allocation hints in the MAIR are ignored. All cacheable accesses are forced to inner read and write allocate. The LDNP and STNP instructions behave the same as the equivalent LDP and STP instructions	0b0

Bits	Name	Description	Reset
[38:37]	ATOM	<p>Atomic instructions execution control. This field defines whether certain cacheable atomic instructions in the Main Manager (MM) are executed near or far.</p> <p>The behavior of the instructions below do not depend on the value of the ATOM field:</p> <ul style="list-style-type: none"> All TCM atomics are executed near. Non-cacheable or Device LLRAM atomics are executed far if CFGLLRAMSHARED=1, otherwise they are executed in the cluster. Cacheable LLRAM atomics are executed far if CFGLLRAMSHARED=1 and the memory location is outer shareable, otherwise they are executed in the cluster. Non-cacheable or Device MM atomics are executed far. Cacheable MM atomics are executed near if BROADCASTOUTERM = 0 or BROADCASTATOMICM = 0. Cacheable MM atomics are executed near if the memory location is non-shareable. Cacheable MM atomics are executed near if the memory location is not aligned to the element size. The MM atomic is executed near if it is a hardware access flag pagetable update. <p>The behavior of the remaining cacheable MM atomics depends on the value of the ATOM field:</p> <p>0b00 Atomic instructions are executed near if they hit in the cache in a unique state, or far if they miss or are shared.</p> <p>0b01 Atomic instructions are executed near.</p> <p>0b10 Most atomic instructions are executed far.</p> <p>0b11 Load atomics, including SWP and CAS, are executed near. Store atomics are executed near if they hit in the cache in a unique state, or far if they miss or are shared.</p>	0b00
[36]	DCWT	<p>L1 data cache cache way tracker control.</p> <p>0b0 Cache way tracker disabled.</p> <p>0b1 Cache way tracker enabled.</p>	0b1
[35:34]	DL2WS	<p>L2 cache Write Streaming no-allocate threshold.</p> <p>0b00 16th consecutive streaming cache line does not allocate in the L2 cache.</p> <p>0b01 128th consecutive streaming cache line does not allocate in the L2 cache.</p> <p>0b10 512th consecutive streaming cache line does not allocate in the L2 cache.</p> <p>0b11 Disables streaming. All write-allocate lines allocate in the L2 cache.</p>	0b01

Bits	Name	Description	Reset
[33:32]	DL1WS	<p>L1 data cache write streaming no-allocate threshold.</p> <p>0b00 4th consecutive streaming cache line does not allocate in the L1 data cache.</p> <p>0b01 64th consecutive streaming cache line does not allocate in the L1 data cache.</p> <p>0b10 128th consecutive streaming cache line does not allocate in the L1 data cache.</p> <p>0b11 Disables write streaming detection. All write-allocate lines allocate in the L1 data cache.</p>	0b00
[31:30]	DPEN	<p>L1 data cache prefetcher enable.</p> <p>0b00 Prefetcher disabled.</p> <p>0b01 Prefetcher enabled without power-aware throttling. May boost performance in some situations, but consumes more power.</p> <p>0b11 Prefetcher enabled with power-aware throttling. May limit performance in some situations compared with DPEN == 0b01, but consumes less power.</p> <p>All other values are Reserved.</p>	0b11
[29:27]	DPDEPTH	<p>L1 data cache prefetcher depth.</p> <p>0b000 Up to 1 outstanding prefetch.</p> <p>0b001 Up to 2 outstanding prefetches.</p> <p>0b010 Up to 3 outstanding prefetches.</p> <p>0b011 Up to 4 outstanding prefetches.</p> <p>0b100 Up to 5 outstanding prefetches.</p> <p>0b101 Up to 6 outstanding prefetches.</p> <p>0b110 Up to 7 outstanding prefetches.</p> <p>0b111 Up to 8 outstanding prefetches.</p>	0b101

Bits	Name	Description	Reset
[26:25]	DPL2	<p>L1 data cache prefetcher into the L2 behavior.</p> <p>0b00 Do not prefetch into the L2.</p> <p>0b01 Initial L2 prefetch is 8 lines ahead of the L1 prefetch.</p> <p>0b10 Initial L2 prefetch is 16 lines ahead of the L1 prefetch.</p> <p>0b11 Initial L2 prefetch is 32 lines ahead of the L1 prefetch.</p>	0b10
[24]	TLAC	<p>Tag Lookup Avoidance Cache (TLAC) enable.</p> <p>0b0 TLAC disabled.</p> <p>0b1 TLAC enabled.</p>	0b1
[23]	DMB	<p>Data Memory Barrier (DMB) instruction behavior.</p> <p>0b0 DMB instructions behave as indicated by the Arm architecture.</p> <p>0b1 DMB instructions behave as DSB instructions (full system barrier).</p> <p>If the value of AArch64-IMP_INTLATENCY_EL2.DMB is not 0 then PE behaves as if the value of the AArch64-IMP_CPUACTLR_EL1.DMB field is 1 for all purposes other than returning the value of a direct read of the field.</p>	0b0
[22:21]	RES0	Reserved	RES0
[20]	WSRST	<p>Write streaming mode reset control. Arm recommends to set this field to 0b1 when accessing memory locations with no Early Write Acknowledgement.</p> <p>0b0 Write streaming mode will not reset when a specific micro-architectural condition is met.</p> <p>0b1 Write streaming mode will reset when a specific micro-architectural condition is met.</p>	0b0
[19]	IREPL	<p>L1 instruction cache replacement policy.</p> <p>0b0 Flexible segregation replacement policy. This replacement policy is designed to reduce pollution caused by excessive speculative fetches from applications with poor locality.</p> <p>0b1 Pseudo-LRU replacement policy.</p>	0b1

Bits	Name	Description	Reset
[18:17]	IPEN	<p>L1 instruction cache prefetcher enable.</p> <p>0b00 Prefetcher disabled.</p> <p>0b01 Prefetcher enabled without power-aware throttling. May boost performance in some situations, but consumes more power.</p> <p>0b11 Prefetcher enabled with power-aware throttling. May limit performance in some situations compared with IPEN == 0b01, but consumes less power.</p> <p>All other values are Reserved.</p>	0b11
[16:15]	IPDEPTH	<p>L1 instruction cache prefetcher depth.</p> <p>0b00 Fetch up to 1 line ahead.</p> <p>0b01 Fetch up to 2 lines ahead.</p> <p>0b10 Fetch up to 3 lines ahead.</p> <p>0b11 Fetch up to 4 lines ahead.</p>	0b10
[14]	IPMODE	<p>L1 instruction cache prefetcher mode.</p> <p>0b0 Early next line mode. The prefetcher aims to always prefetch up to IPDEPTH lines ahead of the current cache line.</p> <p>0b1 Next line mode. The prefetcher begins prefetching up to IPDEPTH lines ahead only when the current line misses in the cache.</p>	0b0
[13]	IMCWT	<p>L1 instruction cache main cache way tracker control. Cache way tracker is used to minimize power on sequential or known accesses to the cache.</p> <p>0b0 Main cache way tracker disabled.</p> <p>0b1 Main cache way tracker enabled.</p>	0b1
[12]	INBTBCWT	<p>L1 instruction cache nano Branch Target Buffer (nBTB) cache way tracker control. Cache way tracker is used to minimize power on sequential or known accesses to the cache.</p> <p>0b0 nBTB cache way tracker disabled.</p> <p>0b1 nBTB cache way tracker enabled.</p>	0b1

Bits	Name	Description	Reset
[11]	ICRSCWT	L1 instruction cache CRS cache way tracker control. Cache way tracker is used to minimize power on sequential or known accesses to the cache. 0b0 CRS cache way tracker disabled. 0b1 CRS cache way tracker enabled.	0b1
[10:9]	IALLOC	L1 instruction cache allocate policy. 0b00 Cache allocates instructions fetched from either MM or LLRAM. 0b01 Cache allocates instructions fetched from MM, but does not allocate instructions fetched from LLRAM. 0b10 Cache allocates instructions fetched from LLRAM, but does not allocate instructions fetched from MM. 0b11 Cache does not allocate instructions fetched from either LLRAM or MM.	0b00
[8]	NBTBFLUSH	Nano Branch Target Buffer (nBTB) flushing behavior. 0b0 nBTB flushes only as needed. 0b1 nBTB flushes as needed, as well as on Context Synchronization, cache maintenance operations, L1I MMS allocations, and parity errors.	0b0
[7]	DGHS	Data Gathering Hint Stalling behavior. 0b0 Upon executing a DGH instruction, the store buffer begins draining the existing slots. The pipeline does not stall. 0b1 Upon executing a DGH instruction, the store buffer begins draining the existing slots. The pipeline stalls until all pending stores to TCMs have drained. No stalling occurs for pending stores to LLRAM or MM.	0b0
[6]	BUBCL	Bubble Closing behavior. This feature is used to improve IPC by making better use of pipelines in some situations. 0b0 The pipeline does not attempt to close bubbles. 0b1 The pipeline closes bubbles when possible.	0b1
[5]	FALF	Fast aligned load forwarding control. This feature is used to improve IPC by making forwarding from aligned load faster in some situations. 0b0 Fast aligned load forwarding disabled. 0b1 Fast aligned load forwarding enabled.	0b1

Bits	Name	Description	Reset
[4]	FPC	Fast pointer chasing control. This feature is used to improve IPC by making load forwarding to base address faster in some situations. 0b0 Fast pointer chasing disabled. 0b1 Fast pointer chasing enabled.	0b1
[3]	OODIV	Out-of-order division control. 0b0 Floating-point divisions complete in-order. 0b1 Floating-point divisions complete out-of-order.	0b1
[2:0]	MI	Multi-issuing control. 0b000 Instructions can only be issued from slot0. 0b010 FP/AdvSIMD instructions can only be issued from slot0. Other instructions can be issued from slot0 or slot1. 0b011 All instructions can be issued from slot0 or slot1. 0b110 FP/AdvSIMD instructions can only be issued from slot0. Other instructions can be issued from slot0, slot1 or slot2. 0b111 All instructions can be issued from slot0, slot1 or slot2. All other values are Reserved.	0b111

Access

MRS <Xt>, IMP_CPUACTLR_EL1

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

MSR IMP_CPUACTLR_EL1, <Xt>

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

Accessibility

MRS <Xt>, IMP_CPUACTLR_EL1

```

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

```

```
else
    X[t, 64] = IMP_CPUACTLR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUACTLR_EL1;
```

MSR IMP_CPUACTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.AUX == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUACTLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CPUACTLR_EL1 = X[t, 64];
```

A.2.2.45 IMP_BPCTLR_EL1, Branch Predictor Control Register

This register controls in which exception levels the branch predictor is active, and whether the dynamic or static branch predictor is used.

It also controls the behavior of the static branch predictor depending on the branch instructions and their condition code. For those fields, the following abbreviations are used:

- T: taken
- NT: not taken

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

R

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx00	xx10	xx10	xx10
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-94: AARCH64_IMP_BPCTLR_EL1 bit assignments

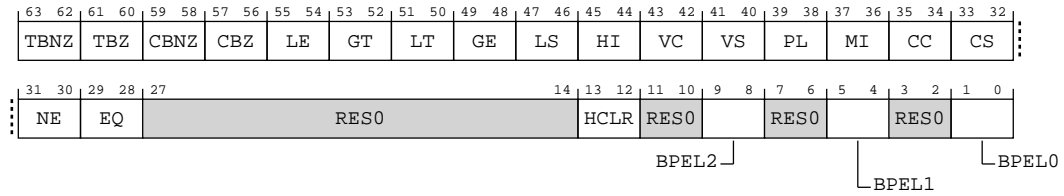


Table A-248: IMP_BPCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:62]	TBNZ	Static branch predictor behavior for TBNZ instructions. 0b00 Predicted NT if offset < 0; predicted NT if offset >= 0. 0b01 Predicted NT if offset < 0; predicted T if offset >= 0. 0b10 Predicted T if offset < 0; predicted NT if offset >= 0. 0b11 Predicted T if offset < 0; predicted T if offset >= 0.	xx
[61:60]	TBZ	Static branch predictor behavior for TBZ instructions. 0b00 Predicted NT if offset < 0; predicted NT if offset >= 0. 0b01 Predicted NT if offset < 0; predicted T if offset >= 0. 0b10 Predicted T if offset < 0; predicted NT if offset >= 0. 0b11 Predicted T if offset < 0; predicted T if offset >= 0.	xx
[59:58]	CBNZ	Static branch predictor behavior for CBNZ instructions. 0b00 Predicted NT if offset < 0; predicted NT if offset >= 0. 0b01 Predicted NT if offset < 0; predicted T if offset >= 0. 0b10 Predicted T if offset < 0; predicted NT if offset >= 0. 0b11 Predicted T if offset < 0; predicted T if offset >= 0.	xx

Bits	Name	Description	Reset
[57:56]	CBZ	<p>Static branch predictor behavior for CBZ instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[55:54]	LE	<p>Static branch predictor behavior for B.LE instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[53:52]	GT	<p>Static branch predictor behavior for B.GT instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[51:50]	LT	<p>Static branch predictor behavior for B.LT instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx

Bits	Name	Description	Reset
[49:48]	GE	<p>Static branch predictor behavior for B.GE instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[47:46]	LS	<p>Static branch predictor behavior for B.LS instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[45:44]	HI	<p>Static branch predictor behavior for B.HI instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[43:42]	VC	<p>Static branch predictor behavior for B.VC instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx

Bits	Name	Description	Reset
[41:40]	VS	<p>Static branch predictor behavior for B.VS instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[39:38]	PL	<p>Static branch predictor behavior for B.PL instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[37:36]	MI	<p>Static branch predictor behavior for B.MI instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[35:34]	CC	<p>Static branch predictor behavior for B.CC instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx

Bits	Name	Description	Reset
[33:32]	CS	<p>Static branch predictor behavior for B.CS instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[31:30]	NE	<p>Static branch predictor behavior for B.NE instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[29:28]	EQ	<p>Static branch predictor behavior for B.EQ instructions.</p> <p>0b00 Predicted NT if offset < 0; predicted NT if offset >= 0.</p> <p>0b01 Predicted NT if offset < 0; predicted T if offset >= 0.</p> <p>0b10 Predicted T if offset < 0; predicted NT if offset >= 0.</p> <p>0b11 Predicted T if offset < 0; predicted T if offset >= 0.</p>	xx
[27:14]	RES0	Reserved	RES0
[13:12]	HCLR	<p>Control the flushing to branch history buffers when taking an exception to a higher exception level.</p> <p>0b00 This control does not cause branch history buffers to be flushed.</p> <p>0b01 Flush the BTAC on the exception entry.</p> <p>0b10 Flush the dynamic branch predictor on the exception entry.</p> <p>0b11 Flush both the dynamic branch predictor and the BTAC on the exception entry.</p>	0b00
[11:10]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[9:8]	BPEL2	<p>Dynamic branch predictor at EL2.</p> <p>0b00 When running at EL2, no branches are predicted.</p> <p>0b01 When running at EL2, branches are predicted with the static predictor.</p> <p>0b10 When running at EL2, branches are predicted with the dynamic predictors.</p> <p>0b11 When running at EL2, branches are predicted with the dynamic predictors, but the BTAC is disabled. Indirect branches which depend on BTAC are not predicted.</p> <p>Other values are Reserved.</p> <p>When PSTATE.EL == EL2 Access to this field is: RW</p> <p>Otherwise Access to this field is: RO</p>	0b10
[7:6]	RES0	Reserved	RES0
[5:4]	BPEL1	<p>Dynamic branch predictor at EL1.</p> <p>0b00 When running at EL1, no branches are predicted.</p> <p>0b01 When running at EL1, branches are predicted with the static predictor.</p> <p>0b10 When running at EL1, branches are predicted with the dynamic predictors.</p> <p>0b11 When running at EL1, branches are predicted with the dynamic predictors, but the BTAC is disabled. Indirect branches which depend on BTAC are not predicted.</p> <p>Other values are Reserved.</p>	0b10
[3:2]	RES0	Reserved	RES0
[1:0]	BPELO	<p>Dynamic branch predictor at ELO.</p> <p>0b00 When running at ELO, no branches are predicted.</p> <p>0b01 When running at ELO, branches are predicted with the static predictor.</p> <p>0b10 When running at ELO, branches are predicted with the dynamic predictors.</p> <p>0b11 When running at ELO, branches are predicted with the dynamic predictors, but the BTAC is disabled. Indirect branches which depend on BTAC are not predicted.</p> <p>Other values are Reserved.</p>	0b10

Access

MRS <Xt>, IMP_BPCTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b11111	0b0001	0b001

MSR IMP_BPCTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b11111	0b0001	0b001

Accessibility

MRS <Xt>, IMP_BPCTLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_BPCTLR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_BPCTLR_EL1;
```

MSR IMP_BPCTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.BPRED == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_BPCTLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_BPCTLR_EL1 = X[t, 64];
```

A.2.2.46 IMP_MEMPROTCTLR_EL1, Memory Protection Control Register

This register controls the memory protection features of the CPU.

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-95: AARCH64_IMP_MEMPROTCTLR_EL1 bit assignments

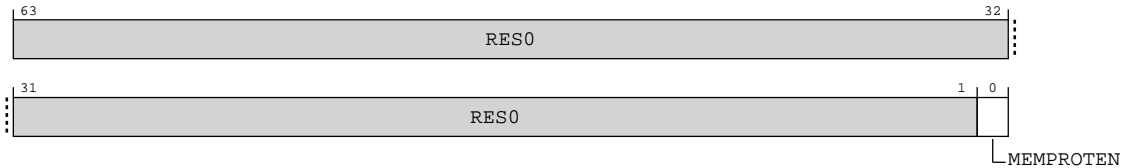


Table A-251: IMP_MEMPROTCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	MEMPROTEN	<p>When RAM_PROTECTION == 1 Memory protection enable.</p> <p>0b0 Structures required to support internal RAM protection functionality are turned off.</p> <p>0b1 Structures required to support internal RAM protection functionality are turned on.</p> <ul style="list-style-type: none"> This field must not be programmed while the caches are enabled, otherwise the behavior of the processor memory system becomes UNPREDICTABLE. Software must ensure that the processor caches are disabled before programming this field. After programming this field, software must ensure that the caches are invalidated before they can be enabled again. If the values of AArch64-IMP_CLUSTERMEMPROTCTLR_EL1.MEMPROTEN and AArch64-IMP_MEMPROTCTLR_EL1.MEMPROTEN for all cores in the processor cluster are not the same, it is possible for wrong data values to be consumed by any core in the cluster, because errors that are locally suppressed by disabling RAM protection may spread through hardware data coherency mechanisms. Arm recommends that AArch64-IMP_CLUSTERMEMPROTCTLR_EL1.MEMPROTEN and all AArch64-IMP_MEMPROTCTLR_EL1.MEMPROTEN fields are programmed with the same value, unless otherwise directed by Arm. <p>Otherwise RES0</p>	<p>When the CFGRAMPROTEN configuration pin is HIGH '1'</p> <p>When the CFGRAMPROTEN configuration pin is LOW '0'</p>

Access

MRS <Xt>, IMP_MEMPROTCTLR_EL1

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

MSR IMP_MEMPROTCTLR_EL1, <Xt>

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

Accessibility

MRS <Xt>, IMP_MEMPROTCTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then

```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_MEMPROTCTLR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_MEMPROTCTLR_EL1;
```

MSR IMP_MEMPROTCTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.DIAGNOSTIC == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_MEMPROTCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    IMP_MEMPROTCTLR_EL1 = X[t, 64];
```

A.2.2.47 IMP_CPUCFR_EL1, CPU Configuration Register

This register shows the value of the (not otherwise software-visible) rendering-time parameters and integration-time pin tie-offs, used to configure this CPU core.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-96: AARCH64_IMP_CPUCFR_EL1 bit assignments

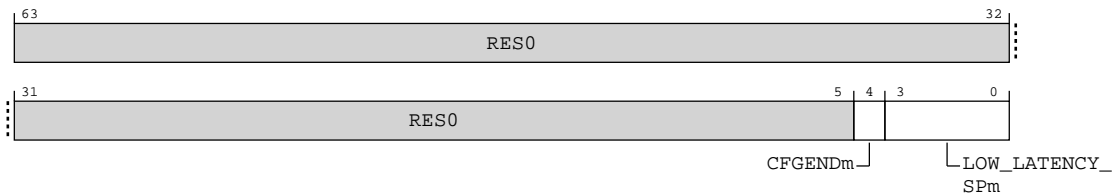


Table A-254: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	RES0
[4]	CFGENDm	Endianness out of reset. This field takes the value of the CFGENDm configuration pin. 0b0 Explicit data accesses and translation table walks set to little-endian out of reset (SCTLR_EL2.EE and SCTLR_EL1.EE fields). 0b1 Explicit data accesses and translation table walks set to big-endian out of reset (SCTLR_EL2.EE and SCTLR_EL1.EE fields).	x
[3:0]	LOW_LATENCY_Spm	Low-latency Single-Precision Floating-Point. This field takes the value of the per-core LOW_LATENCY_Spm parameter. 0b0000 The core does not implement any special hardware for SP FP operations. 0b0001 The core implements additional hardware that supports 3-cycle non-vector SP FP operations.	xxxx

Access

MRS <Xt>, IMP_CPUCFR_EL1

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

Accessibility

MRS <Xt>, IMP_CPUCFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUCFR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUCFR_EL1;
```

A.2.2.48 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

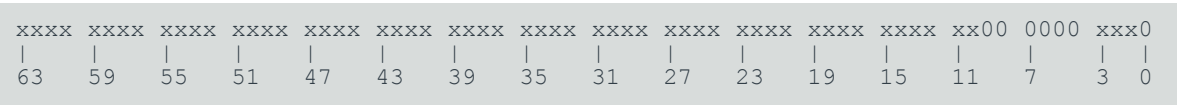
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-97: AARCH64_IMP_CPUPWRCTLR_EL1 bit assignments

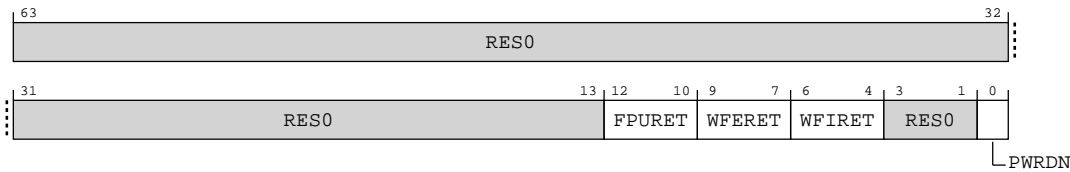


Table A-256: IMP_CPUPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[12:10]	FPURET	<p>When NEON_FPm > 0</p> <p>FPU retention control.</p> <p>0b000 FPU retention disabled.</p> <p>0b001 When the FPU is idle, request transition to FUNC_RET after 2 timer ticks (40 ns - 200 ns).</p> <p>0b010 When the FPU is idle, request transition to FUNC_RET after 8 timer ticks (160 ns - 800 ns).</p> <p>0b011 When the FPU is idle, request transition to FUNC_RET after 32 timer ticks (640 ns - 3.2 us).</p> <p>0b100 When the FPU is idle, request transition to FUNC_RET after 64 timer ticks (1.28 us - 6.4 us).</p> <p>0b101 When the FPU is idle, request transition to FUNC_RET after 128 timer ticks (2.56 us - 12.8 us).</p> <p>0b110 When the FPU is idle, request transition to FUNC_RET after 256 timer ticks (5.12 us - 25.6 us).</p> <p>0b111 When the FPU is idle, request transition to FUNC_RET after 512 timer ticks (10.24 us - 51.2 us).</p> <p>The time range estimations are given assuming an architectural timer clock frequency between 50 MHz - 10 MHz.</p> <p>Otherwise RES0</p>	'000'

Bits	Name	Description	Reset
[9:7]	WFERET	<p>Wait for Event retention control.</p> <p>0b000 Core retention disabled for WFE.</p> <p>0b001 When the core enters WFE state, request transition to FULL_RET after 2 timer ticks (40 ns - 200 ns).</p> <p>0b010 When the core enters WFE state, request transition to FULL_RET after 8 timer ticks (160 ns - 800 ns).</p> <p>0b011 When the core enters WFE state, request transition to FULL_RET after 32 timer ticks (640 ns - 3.2 us).</p> <p>0b100 When the core enters WFE state, request transition to FULL_RET after 64 timer ticks (1.28 us - 6.4 us).</p> <p>0b101 When the core enters WFE state, request transition to FULL_RET after 128 timer ticks (2.56 us - 12.8 us).</p> <p>0b110 When the core enters WFE state, request transition to FULL_RET after 256 timer ticks (5.12 us - 25.6 us).</p> <p>0b111 When the core enters WFE state, request transition to FULL_RET after 512 timer ticks (10.24 us - 51.2 us).</p> <p>The time range estimations are given assuming an architectural timer clock frequency between 50 MHz - 10 MHz.</p>	0b000

Bits	Name	Description	Reset
[6:4]	WFIRET	<p>Wait for Interrupt retention control.</p> <p>0b000 Core retention disabled for WFI.</p> <p>0b001 When the core enters WFI state, request transition to FULL_RET after 2 timer ticks (40 ns - 200 ns).</p> <p>0b010 When the core enters WFI state, request transition to FULL_RET after 8 timer ticks (160 ns - 800 ns).</p> <p>0b011 When the core enters WFI state, request transition to FULL_RET after 32 timer ticks (640 ns - 3.2 us).</p> <p>0b100 When the core enters WFI state, request transition to FULL_RET after 64 timer ticks (1.28 us - 6.4 us).</p> <p>0b101 When the core enters WFI state, request transition to FULL_RET after 128 timer ticks (2.56 us - 12.8 us).</p> <p>0b110 When the core enters WFI state, request transition to FULL_RET after 256 timer ticks (5.12 us - 25.6 us).</p> <p>0b111 When the core enters WFI state, request transition to FULL_RET after 512 timer ticks (10.24 us - 51.2 us).</p> <p>The time range estimations are given assuming an architectural timer clock frequency between 50 MHz - 10 MHz.</p>	0b000
[3:1]	RES0	Reserved	RES0
[0]	PWRDN	<p>Indicates to the power controller if the core wants to power down when it enters WFI state.</p> <p>0b0 When the core enters WFI state, do not request to power down.</p> <p>0b1 When the core enters WFI state, request to power down.</p>	0b0

Access

MRS <Xt>, IMP_CPUPWRCTLR_EL1

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

MSR IMP_CPUPWRCTLR_EL1, <Xt>

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

Accessibility

MRS <Xt>, IMP_CPUPWRCTLR_EL1

```

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

```

MSR IMP_CPUPWRCTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.POWER == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    IMP_CPUPWRCTLR_EL1 = X[t, 64];

```

A.2.2.49 IMP_CLUSTERCFR_EL1, Cluster Configuration Register

This register shows the value of the (not otherwise software-visible) rendering-time parameters and integration-time pin tie-offs, used to configure the processor cluster.

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-98: AARCH64_IMP_CLUSTERCFR_EL1 bit assignments

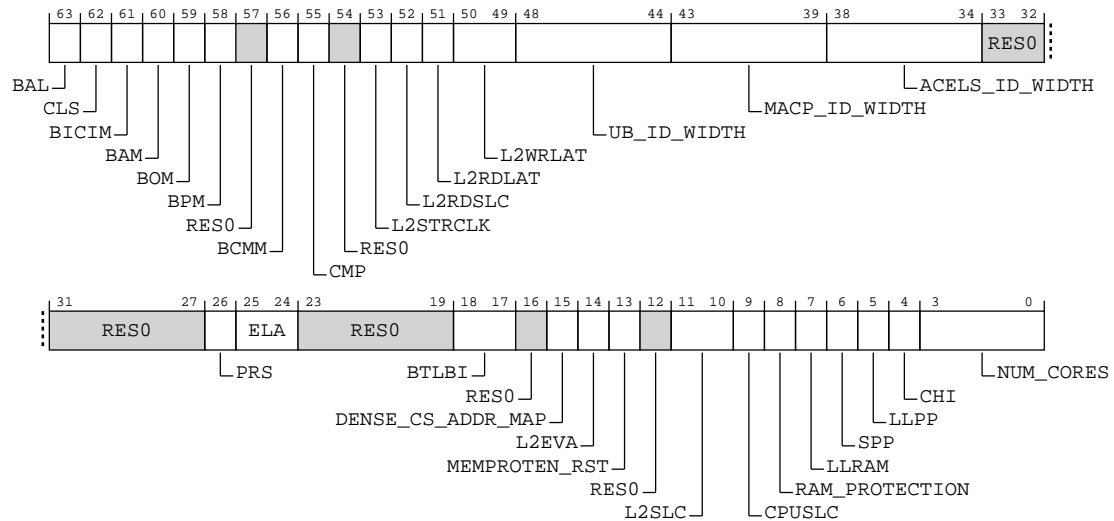


Table A-259: IMP_CLUSTERCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	BAL	Broadcast Atomic LLRAM. This field takes the value of the BROADCASTATOMICL configuration pin. 0b0 Atomic transactions on the LLRAM port are not sent externally. 0b1 Atomic transactions on the LLRAM port are sent externally.	x
[62]	CLS	Configuration LLRAM Shared. This field takes the value of the CFGLLRAMSHARED configuration pin. 0b0 Shareable transactions on the LLRAM port are not broadcast externally. 0b1 Shareable transactions on the LLRAM port are broadcast externally.	x

Bits	Name	Description	Reset
[61]	BICIM	<p>When CHI == 1 Broadcast Instruction Cache Invalidate Main Manager. This field takes the value of the BROADCASTICINVALM configuration pin.</p> <p>0b0 Instruction cache DVM transactions on the Main Manager port are not sent externally.</p> <p>0b1 Instruction cache DVM transactions on the Main Manager port are sent externally.</p> <p>Otherwise RES0</p>	x
[60]	BAM	<p>Broadcast Atomic Main Manager. This field takes the value of the BROADCASTATOMICM configuration pin.</p> <p>0b0 Atomic transactions on the Main Manager port are not sent externally.</p> <p>0b1 Atomic transactions on the Main Manager port are sent externally.</p>	x
[59]	BOM	<p>When CHI == 1 Broadcast Outer Main Manager. This field takes the value of the BROADCASTOUTERM configuration pin.</p> <p>0b0 Shareable transactions on the Main Manager port are not broadcast externally.</p> <p>0b1 Shareable transactions on the Main Manager port are broadcast externally.</p> <p>Otherwise RES0</p>	x
[58]	BPM	<p>When CHI == 1 Broadcast Persist Maintenance Main Manager. This field takes the value of the BROADCASTPERSISTM configuration pin.</p> <p>0b0 On the Main Manager port, DC CVAP instructions are treated the same as DC CVAC.</p> <p>0b1 On the Main Manager port, DC CVAP instructions are sent as a clean to the point of persistence transactions.</p> <p>Otherwise RES0</p>	x
[57]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[56]	BCMM	<p>When CHI == 1</p> <p>Broadcast Cache Maintenance Main Manager. This field takes the value of the BROADCASTCACHEMAINTM configuration pin.</p> <p>0b0</p> <p>Cache maintenance operations are not broadcast to downstream caches on the Main Manager port.</p> <p>0b1</p> <p>Cache maintenance operations are broadcast to downstream caches on the Main Manager port.</p> <p>Otherwise</p> <p>RES0</p>	x
[55]	CMP	<p>Main Manager port poison support. This field takes the value of the CFGMMPOISON configuration pin.</p> <p>0b0</p> <p>Main Manager port does not support poison.</p> <p>0b1</p> <p>Main Manager port supports poison.</p>	x
[54]	RES0	Reserved	RES0
[53]	L2STRCLK	<p>L2 cache Stretch Clock. This field takes the value of the L2_DATA_STRETCH_CLK global parameter.</p> <p>0b0</p> <p>L2 data RAMs clock not stretched. The clock pulse will be high for half an SCLK cycle.</p> <p>0b1</p> <p>L2 data RAMs clock stretched. The clock pulse will be high for a whole SCLK cycle.</p>	x
[52]	L2RDSLCL	<p>L2 cache Read Slice. This field takes the value of the L2_DATA_RD_SLICE global parameter.</p> <p>0b0</p> <p>No register slice present for reading the L2 data RAMs.</p> <p>0b1</p> <p>Register slice present for reading the L2 data RAMs.</p>	x
[51]	L2RDLAT	<p>L2 cache Read Latency. This field takes the value of the L2_DATA_RD_LATENCY global parameter.</p> <p>0b0</p> <p>2 cycles output delay from L2 data RAMs.</p> <p>0b1</p> <p>3 cycles output delay from L2 data RAMs.</p>	x
[50:49]	L2WRLAT	<p>L2 cache Write Latency. This field takes the value of the L2_DATA_WR_LATENCY global parameter.</p> <p>0b00</p> <p>1 cycle input delay from L2 data RAMs.</p> <p>0b01</p> <p>2 cycles input delay from L2 data RAMs.</p> <p>0b10</p> <p>2 cycles input delay plus 1 cycle hold from L2 data RAMs.</p>	xx

Bits	Name	Description	Reset
[48:44]	UB_ID_WIDTH	Utility Bus port AXI ID Width. This field takes the value of the UB_ID_WIDTH global parameter.	5 {x}
[43:39]	MACP_ID_WIDTH	MACP AXI ID Width. This field takes the value of the MACP_ID_WIDTH global parameter.	5 {x}
[38:34]	ACELS_ID_WIDTH	ACELS port AXI ID Width. This field takes the value of the ACELS_ID_WIDTH global parameter.	5 {x}
[33:27]	RES0	Reserved	RES0
[26]	PRS	PPU Reset State. This field takes the value of the PPU_RST_STATE global parameter. 0b0 Power Policy Units stay in OFF state after reset. 0b1 Power Policy Units transition to ON after reset.	x
[25:24]	ELA	Embedded Logic Analyzer. This field depends on the value of the ELA global parameter and the ELADISABLE configuration pin. 0b00 The processor does not implement any embedded logic analyzers. 0b01 The processor is implemented with embedded CoreSight ELA-600 embedded logic analyzers. The analyzers are disabled and cannot be used. 0b11 The processor is implemented with embedded CoreSight ELA-600 embedded logic analyzers. The analyzers are available to be used.	xx
[23:19]	RES0	Reserved	RES0
[18:17]	BTLBI	When CHI == 1 Broadcast TLBI Main Manager. BTLBI[0] takes the value of the BROADCASTTLBIINNERM configuration pin and BTLBI[1] takes the value of the BROADCASTTLBIOUTERM configuration pin. 0b00 No TLBI operations are broadcast outside the cluster. 0b01 Illegal configuration. 0b10 Out Shareable TLBI operations are broadcast outside the cluster. 0b11 Both Inner Shareable TLBI operations and Out Shareable TLBI operations are broadcast outside the cluster. Otherwise RES0	xx
[16]	RES0	Reserved	RES0
[15]	DENSE_CS_ADDR_MAP	Dense memory map for Utility Bus and Debug port. This field takes the value of the DENSE_CS_ADDR_MAP global parameter. 0b0 Processor implements the sparse memory map. 0b1 Processor implements the dense memory map.	x

Bits	Name	Description	Reset
[14]	L2EVA	<p>L2 cache POP RAM allocate evict behavior. This field takes the value of the CFGL2EVAIMP configuration pin.</p> <p>0b0</p> <p>If using POP RAMs, processor does not implement the optimized read/write allocate evict mechanism.</p> <p>0b1</p> <p>If using POP RAMs, processor is implemented with the optimized read/write allocate evict mechanism.</p>	x
[13]	MEMPROTEN_RST	<p>Reset values of AArch64-IMP_CLUSTERMEMPROTCTLR_EL1.MEMPROTEN and AArch64-IMP_MEMPROTCTLR_EL1.MEMPROTEN. This field takes the value of the CFGGRAMPROTEN configuration pin.</p> <p>0b0</p> <p>Reset values of AArch64-IMP_CLUSTERMEMPROTCTLR_EL1.MEMPROTEN and AArch64-IMP_MEMPROTCTLR_EL1.MEMPROTEN are 0b0.</p> <p>0b1</p> <p>Reset values of AArch64-IMP_CLUSTERMEMPROTCTLR_EL1.MEMPROTEN and AArch64-IMP_MEMPROTCTLR_EL1.MEMPROTEN are 0b1.</p>	x
[12]	RES0	Reserved	RES0
[11:10]	L2SLC	<p>L2 Slices and RAM partitions. This field takes the value of the L2_SLICES global parameter.</p> <p>0b01</p> <p>The L2 cache implements a single slice and a single RAM partition.</p> <p>0b10</p> <p>The L2 cache implements two slices and two RAM partitions.</p>	xx
[9]	CPUSLC	<p>Extra register slices between the cores and the cluster logic. This field takes the value of the CPU_SLICE global parameter.</p> <p>0b0</p> <p>No additional register slices exist between the cores and cluster logic.</p> <p>0b1</p> <p>One additional register slice exists between the cores and cluster logic.</p>	x
[8]	RAM_PROTECTION	<p>Core RAM protection. This field takes the value of the RAM_PROTECTION global parameter.</p> <p>0b0</p> <p>The processor does not implement any structures to protect internal RAMs.</p> <p>0b1</p> <p>The processor is implemented with structures required to support internal RAM protection functionality.</p>	x
[7]	LLRAM	<p>LLRAM interface. This field takes the value of the LLRAM global parameter.</p> <p>0b0</p> <p>LLRAM interface is not implemented, and associated hardware logic is not present.</p> <p>0b1</p> <p>LLRAM interface is implemented.</p>	x

Bits	Name	Description	Reset
[6]	SPP	SPP interface. This field takes the value of the SPP global parameter. 0b0 SPP interface is not implemented, and associated hardware logic is not present. 0b1 SPP interface is implemented.	x
[5]	LLPP	LLPP interface. This field takes the value of the LLPP global parameter. 0b0 LLPP interface is not implemented, and associated hardware logic is not present. 0b1 LLPP interface is implemented.	x
[4]	CHI	Main Manager port type. This field takes the value of the CHI global parameter. 0b0 The MM port implements an AXI5 interface. 0b1 The MM port implements a CHI.E interface.	x
[3:0]	NUM_CORES	Number of CPU cores in the cluster. This field takes the value of the NUM_CORES global parameter. 0b0001 1 CPU core implemented in the cluster. 0b0010 2 CPU cores implemented in the cluster. 0b0011 3 CPU cores implemented in the cluster. 0b0100 4 CPU cores implemented in the cluster. 0b0101 5 CPU cores implemented in the cluster. 0b0110 6 CPU cores implemented in the cluster. 0b0111 7 CPU cores implemented in the cluster. 0b1000 8 CPU cores implemented in the cluster.	xxxx

Access

MRS <Xt>, IMP_CLUSTERCFR_EL1

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

Accessibility

MRS <Xt>, IMP_CLUSTERCFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERCFR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CLUSTERCFR_EL1;
```

A.2.2.50 IMP_CLUSTERACTLR_EL1, Cluster Auxiliary Control Register

This register contains control bits that affect the cluster 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	xxx0	xx11	xxx1	xxxx	xx00	0001	xxxx	xx00	001x
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-99: AARCH64_IMP_CLUSTERACTLR_EL1 bit assignments

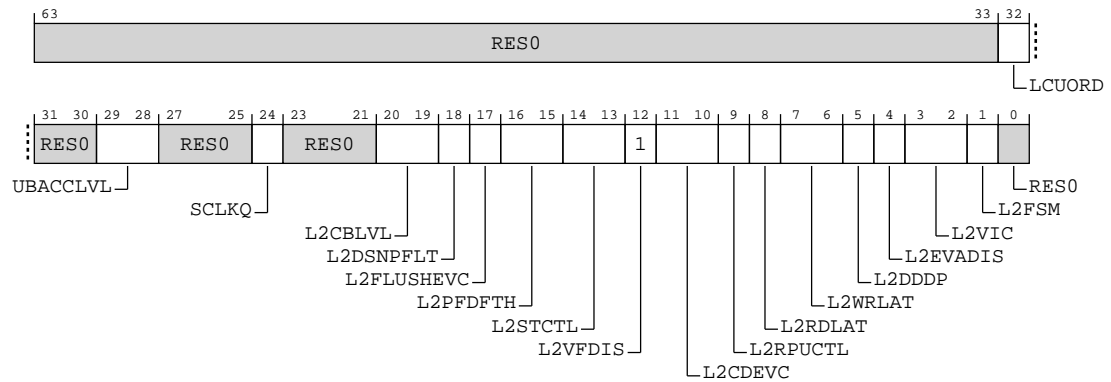


Table A-261: IMP_CLUSTERACTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:33]	RES0	Reserved	RES0
[32]	LCUORD	LCU Ordering behavior. 0b0 Normal behavior. 0b1 Force ordering. All memory effects seen in the order they are arbitrated.	0b0
[31:30]	RES0	Reserved	RES0
[29:28]	UBACCLVL	Access level from the Utility bus. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ⁷
[27:25]	RES0	Reserved	RES0
[24]	SCLKQ	SCLK Q-channel operational. 0b0 The cluster SCLK Q-channel is permanently active. The SCLKQACTIVE output is always HIGH. Software can use this setting to improve latency in some cases when all cores are in WFx state, or for incoming requests from the ACELS and MACP ports. 0b1 The cluster SCLK Q-channel operates normally.	0b1
[23:21]	RES0	Reserved	RES0

⁷ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.

Bits	Name	Description	Reset
[20:19]	L2CBLVL	<p>L2 internal CBUSY generation control. Controls the level of activity within the L2 before prefetch throttling is applied.</p> <p>This field resets to an appropriate value depending on the number of cores in the cluster:</p> <ul style="list-style-type: none"> When 1 core is present, L2CBLVL resets to 0b00. When 2-7 cores are present, L2CBLVL resets to 0b01. When 8 cores are present, L2CBLVL resets to 0b11. <p>0b00 Feedback is disabled, no prefetch throttling will occur.</p> <p>0b01 Normal thresholds are used, prefetch throttling may occur.</p> <p>0b10 Conservative threshold are used, prefetch throttling may occur earlier than with normal thresholds.</p> <p>0b11 Most conservative thresholds are used, prefetch throttling may occur earlier than with conservative thresholds.</p>	xx
[18]	L2DSNPFLT	<p>When CHI == 1 L2 downstream snoop filter present.</p> <p>0b0 Disables sending Evict transactions when clean cache lines are evicted without data.</p> <p>0b1 Enables sending Evict transactions when clean cache lines are evicted without data.</p> <p>Otherwise RES0</p>	'0'
[17]	L2FLUSHEVC	<p>L2 flush evict control.</p> <p>0b0 Disables sending data when hardware cache flushes or DC C1SW instructions evict a clean cache line.</p> <p>0b1 Sending data when hardware cache flushes or DC C1SW instructions evict clean cache lines is controlled by L2CDEVC. Sending of Evict transactions is controlled by L2DSNPFLT.</p>	0b0
[16:15]	L2PFDFTH	<p>L2 prefetch data forwarding threshold.</p> <p>0b00 Default prefetch forwarding behavior.</p> <p>0b01 Faster prefetch forwarding timeout.</p> <p>0b10 Immediate prefetch forwarding timeout (no waiting).</p> <p>0b11 Prefetch forwarding is disabled.</p>	0b00

Bits	Name	Description	Reset
[14:13]	L2STCTL	<p>L2 cache stashing control.</p> <p>0b00</p> <p>Stashes targeting L2 cache will allocate as if the line were brought in by a load.</p> <p>0b01</p> <p>Stashes targeting L2 cache will allocate and be marked as preferred targets for eviction.</p> <p>0b10</p> <p>Stashes targeting L2 cache will allocate as if the line were brought in by a load, but will only allocate to odd numbered cache ways.</p> <p>0b11</p> <p>Stashes targeting L2 cache will be ignored.</p>	0b00
[12]	L2VFDIS	<p>L2 allocation control for L1 evictions.</p> <p>0b1</p> <p>L1 victims will be allocated to the L2 cache before they can reallocate to L1.</p>	0b1
[11:10]	L2CDEV	<p>When CHI == 1</p> <p>L2 cache flushing behavior.</p> <p>0b00</p> <p>Disables sending data when clean cache lines are evicted.</p> <p>0b01</p> <p>Enables sending WriteEvictFull transactions when Unique Clean cache lines are evicted. Shared Clean cache line evictions do not send data.</p> <p>0b10</p> <p>Enables sending WriteEvictOrEvict transactions when Unique Clean cache lines are evicted. Shared Clean cache line evictions do not send data.</p> <p>0b11</p> <p>Enables sending WriteEvictOrEvict transactions when Unique Clean or Shared Clean cache lines are evicted.</p> <p>Otherwise</p> <p>L2 cache flushing behavior.</p> <p>0b00</p> <p>Disables sending data when clean cache lines are evicted.</p> <p>0b11</p> <p>Enables sending data when clean cache lines are evicted.</p> <p>All other values are reserved.</p>	'00'

Bits	Name	Description	Reset
[9]	L2RPUCTL	<p>When CHI == 1</p> <p>L2 ReadPreferUnique control.</p> <p>0b0</p> <p>ReadPreferUnique will not be sent outside the cluster when core ReadPreferUnique hits SharedClean inside the cluster.</p> <p>0b1</p> <p>ReadPreferUnique will be sent outside the cluster when core ReadPreferUnique hits SharedClean inside the cluster.</p> <p>Otherwise</p> <p>RES0</p>	'0'
[8]	L2RDLAT	<p>L2 cache Read Latency.</p> <p>This field resets to the value of the L2_DATA_RD_LATENCY global parameter, visible in AArch64-IMP_CLUSTERCFR_EL1.L2RDLAT.</p> <p>This field can be updated by software after a cluster reset, but will only take effect as soon as the L2 cache is idle. Arm recommends that changes are done in periods with low expected activity, for example when the system is booting and before the L2 cache is enabled.</p> <p>0b0</p> <p>2 cycles output delay from L2 data RAMs.</p> <p>0b1</p> <p>3 cycles output delay from L2 data RAMs.</p>	x
[7:6]	L2WRLAT	<p>L2 cache Write Latency.</p> <p>This field resets to the value of the L2_DATA_WR_LATENCY global parameter, visible in AArch64-IMP_CLUSTERCFR_EL1.L2WRLAT.</p> <p>This field can be updated by software after a cluster reset, but will only take effect as soon as the L2 cache is idle. Arm recommends that changes are done in periods with low expected activity, for example when the system is booting and before the L2 cache is enabled.</p> <p>0b00</p> <p>1 cycle input delay from L2 data RAMs.</p> <p>0b01</p> <p>2 cycles input delay from L2 data RAMs.</p> <p>0b10</p> <p>2 cycles input delay plus 1 cycle hold from L2 data RAMs.</p>	xx
[5]	L2DDDP	<p>L2 cache stashing snoop disable datapull.</p> <p>0b0</p> <p>A stashing snoop will send a datapull request.</p> <p>0b1</p> <p>A stashing snoop will not send a datapull request.</p>	0b0

Bits	Name	Description	Reset
[4]	L2EVADIS	L2 cache POP RAM allocate evict behavior. 0b0 If using POP RAMs, enable the optimized read/write allocate evict mechanism. 0b1 If using POP RAMs, disable the optimized read/write allocate evict mechanism. If the CFGL2EVAIMP configuration pin is 0b0, this bit is RAO/WI .	0b0
[3:2]	L2VIC	L2 Victim control. 0b00 I-side requests use new age of 0b10 (less likely to be evicted). D-side requests use new age of 0b01 (more likely to be evicted). 0b01 Both i-side and d-side requests use 0b01. 0b10 Both i-side and d-side requests use 0b10. 0b11 Disabled. All ages are treated the same.	0b00
[1]	L2FSM	L2 hazarding optimisation control. Hazarding optimisations will improve average performance but may reduce determinism. 0b0 Hazarding optimisations disabled. 0b1 Hazarding optimisations enabled.	0b1
[0]	RES0	Reserved	RES0

Access

MRS <Xt>, IMP_CLUSTERACTLR_EL1

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

MSR IMP_CLUSTERACTLR_EL1, <Xt>

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

Accessibility

MRS <Xt>, IMP_CLUSTERACTLR_EL1

```

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

```
else
    X[t, 64] = IMP_CLUSTERACTLR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERACTLR_EL1;
```

MSR IMP_CLUSTERACTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.AUX == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERACTLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERACTLR_EL1 = X[t, 64];
```

A.2.2.51 IMP_CLUSTERPWRCTLR_EL1, Cluster Power Control Register

This register controls power features of the cluster. There is only one IMP_CLUSTERPWRCTLR_EL1 register implemented in the cluster, which all cores can access.

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-100: AARCH64_IMP_CLUSTERPWRCTLR_EL1 bit assignments

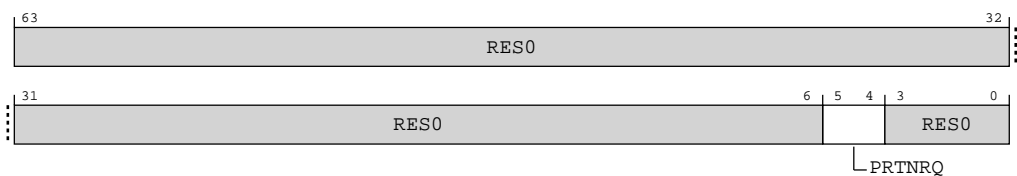


Table A-264: IMP_CLUSTERPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:6]	RES0	Reserved	RES0
[5:4]	PRTNRQ	When L2_SLICES == 2 L2 cache Partition power request. These bits are passed to the PPU as an advisory request for which partitions to power. 0b00 Request that none of the L2 cache partitions are powered up. 0b01 Request that half of the L2 cache is powered up. 0b11 Request that both partitions of the L2 cache are powered up. Otherwise L2 cache Partition power request. These bits are passed to the PPU as an advisory request for which partitions to power. 0b00 Request that none of the L2 cache partitions are powered up. 0b11 Request that both partitions of the L2 cache are powered up.	'11' ⁸
[3:0]	RES0	Reserved	RES0

Access

MRS <Xt>, IMP_CLUSTERPWRCTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0011	0b101

MSR IMP_CLUSTERPWRCTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0011	0b101

⁸ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.

Copyright © 2022–2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Accessibility

MRS <Xt>, IMP_CLUSTERPWRCTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPWRCTLR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CLUSTERPWRCTLR_EL1;

```

MSR IMP_CLUSTERPWRCTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.POWER == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPWRCTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        IMP_CLUSTERPWRCTLR_EL1 = X[t, 64];

```

A.2.2.52 IMP_CLUSTERPWRDN_EL1, Cluster Power Down Register

This register controls powerdown requirements of the cluster. This register is banked per core, i.e. each core accesses its own copy of the register. Cluster logic combines all registers contents to decide the power strategy.

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	xx00
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-101: AARCH64_IMP_CLUSTERPWRDN_EL1 bit assignments

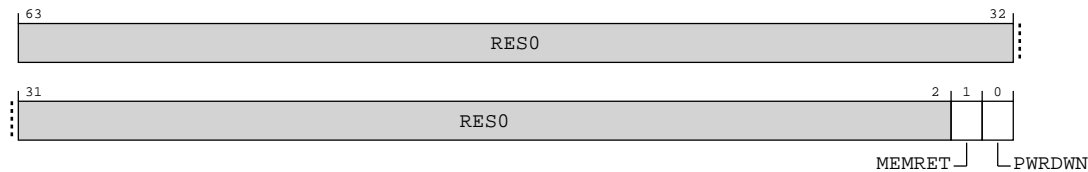


Table A-267: IMP_CLUSTERPWRDN_EL1 bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	RES0
[1]	MEMRET	Memory Retention request. 0b0 No indication to the PPU. 0b1 Indicate to the PPU that memory retention is desired when all cores are powered down. This is an advisory status to the PPU, and will not cause an explicit request to power off the cluster to be denied.	0b0 ⁹
[0]	PWRDWN	Power Down request. 0b0 No indication to the PPU. 0b1 Indicate to the PPU that cluster power is required even when all cores are powered down. This is an advisory status to the PPU, and will not cause an explicit request to power off the cluster to be denied.	0b0

Access

MRS <Xt>, IMP_CLUSTERPWRDN_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0011	0b110

MSR IMP_CLUSTERPWRDN_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0011	0b110

⁹ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.

Accessibility

MRS <Xt>, IMP_CLUSTERPWRDN_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPWRDN_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CLUSTERPWRDN_EL1;

```

MSR IMP_CLUSTERPWRDN_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.POWER == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPWRDN_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        IMP_CLUSTERPWRDN_EL1 = X[t, 64];

```

A.2.2.53 IMP_CLUSTERPWRSTAT_EL1, Cluster Power Status Register

This register contains the current status of the cluster power features. There is only one IMP_CLUSTERPWRSTAT_EL1 register implemented in the cluster, which all cores can access.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

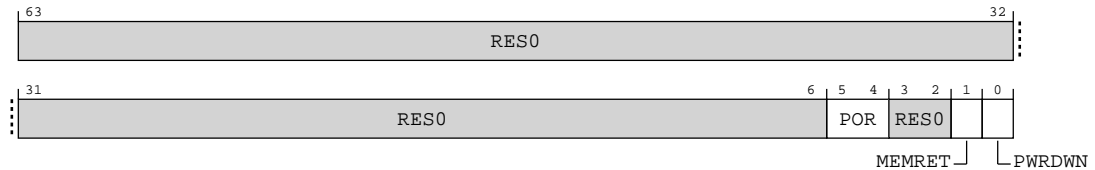
Reset value

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-102: AARCH64_IMP_CLUSTERPWRSTAT_EL1 bit assignments**Table A-270: IMP_CLUSTERPWRSTAT_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:6]	RES0	Reserved	RES0
[5:4]	POR	<p>When L2_SLICES == 2</p> <p>L2 cache Partition power status. This indicates which cache partitions are currently powered up and available. It can be used to determine when the state requested in the AArch64-IMP_CLUSTERPWRCTLR_EL1.PRTNRQ has taken effect.</p> <p>0b00</p> <p>None of the L2 cache partitions are powered up.</p> <p>0b01</p> <p>Half of the L2 cache is powered up.</p> <p>0b11</p> <p>Both partitions of the L2 cache are powered up.</p> <p>Otherwise</p> <p>L2 cache Partition power status. This indicates which cache partitions are currently powered up and available. It can be used to determine when the state requested in the AArch64-IMP_CLUSTERPWRCTLR_EL1.PRTNRQ has taken effect.</p> <p>0b00</p> <p>None of the L2 cache partitions are powered up.</p> <p>0b11</p> <p>Both partitions of the L2 cache are powered up.</p>	xx
[3:2]	RES0	Reserved	RES0
[1]	MEMRET	<p>Memory retention enabled. This bit is a combined version of all banked per-thread bits from the AArch64-IMP_CLUSTERPWRDN_EL1 register.</p> <p>0b0</p> <p>Memory retention inactive.</p> <p>0b1</p> <p>Memory retention enabled when all cores powered down.</p>	x

Bits	Name	Description	Reset
[0]	PWRDWN	Disabled cluster power down. Note this bit is a combined version of all banked per-thread bits from the AArch64-IMP_CLUSTERPWRDN_EL1 register. 0b0 Cluster power down active. 0b1 Cluster power down disabled when all cores powered down.	x

Access

MRS <Xt>, IMP_CLUSTERPWRSTAT_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0011	0b111

Accessibility

MRS <Xt>, IMP_CLUSTERPWRSTAT_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPWRSTAT_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPWRSTAT_EL1;
```

A.2.2.54 IMP_CLUSTERSID_EL1, Cluster Scheme ID Register

This register controls the L2 cache partitioning Scheme ID for a core. The register is banked per core, i.e. each core accesses its own copy of the register.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

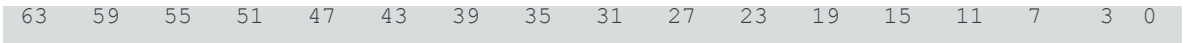
Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x000



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-103: AARCH64_IMP_CLUSTERSID_EL1 bit assignments

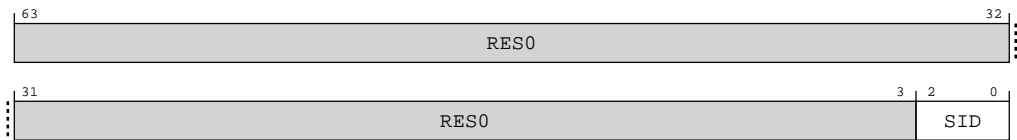


Table A-272: IMP_CLUSTERSID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2:0]	SID	Scheme ID. Selects one of the 8 available L2 cache scheme IDs. Memory accesses from this core which reach the L2 cache will use the scheme ID specified by the SID field, as defined by the AArch64-IMP_CLUSTERPARTCR_EL1 register.	0b000

Access

MRS <Xt>, IMP_CLUSTERSID_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b000

MSR IMP_CLUSTERSID_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b000

Accessibility

MRS <Xt>, IMP_CLUSTERSID_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERSID_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CLUSTERSID_EL1;
```

MSR IMP_CLUSTERSID_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPORTS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERSID_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERSID_EL1 = X[t, 64];
```

A.2.2.55 IMP_CLUSTERACPSID_EL1, Cluster ACP Scheme ID Register

This register controls the L2 cache partitioning Scheme IDs for ACP accesses and Stash requests.

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	x000	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-104: AARCH64_IMP_CLUSTERACPSID_EL1 bit assignments

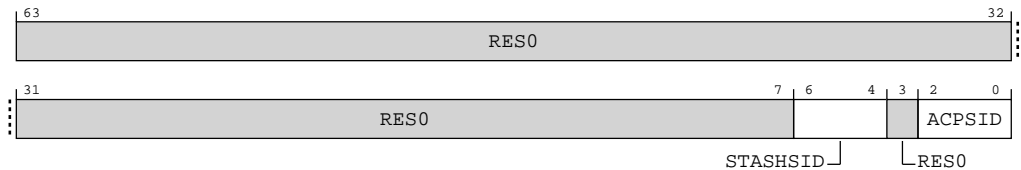


Table A-275: IMP_CLUSTERACPSID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:7]	RES0	Reserved	RES0
[6:4]	STASHID	Stash Scheme ID. Selects one of the 8 available L2 cache scheme IDs. Stash requests from the MM port (when using CHI) which reach the L2 cache will use the scheme ID specified by the STASHID field, as defined by the AArch64-IMP_CLUSTERPARTCR_EL1 register. This field has no effect when CHI is not implemented.	0b000
[3]	RES0	Reserved	RES0
[2:0]	ACPSID	ACP Scheme ID. Selects one of the 8 available L2 cache scheme IDs. Accesses from the MACP port which reach the L2 cache will use the scheme ID specified by the ACPSID field, as defined by the AArch64-IMP_CLUSTERPARTCR_EL1 register.	0b000

Access

MRS <Xt>, IMP_CLUSTERACPSID_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b001

MSR IMP_CLUSTERACPSID_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b001

Accessibility

MRS <Xt>, IMP_CLUSTERACPSID_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERACPSID_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERACPSID_EL1;

```

MSR IMP_CLUSTERACPSID_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPORTS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERACPSID_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    IMP_CLUSTERACPSID_EL1 = X[t, 64];

```

A.2.2.56 IMP_CLUSTERPARTCR_EL1, Cluster Partition Control Register

This register controls the L2 cache partitioning. It defines how the 4 way groups map onto the 8 Scheme IDs.

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	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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-105: AARCH64_IMP_CLUSTERPARTCR_EL1 bit assignments

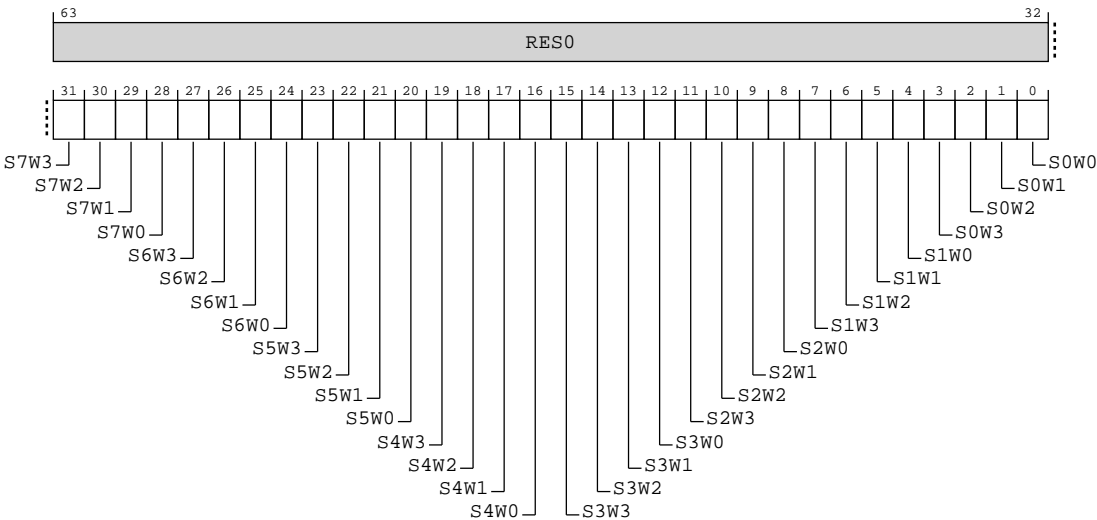


Table A-278: IMP_CLUSTERPARTCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	S7W3	L2 Cache Way Group 3 mapping to Scheme ID 7. 0b0 Way Group 3 not assigned to Scheme ID 7. 0b1 Way Group 3 assigned to Scheme ID 7.	0b0
[30]	S7W2	L2 Cache Way Group 2 mapping to Scheme ID 7. 0b0 Way Group 2 not assigned to Scheme ID 7. 0b1 Way Group 2 assigned to Scheme ID 7.	0b0
[29]	S7W1	L2 Cache Way Group 1 mapping to Scheme ID 7. 0b0 Way Group 1 not assigned to Scheme ID 7. 0b1 Way Group 1 assigned to Scheme ID 7.	0b0
[28]	S7W0	L2 Cache Way Group 0 mapping to Scheme ID 7. 0b0 Way Group 0 not assigned to Scheme ID 7. 0b1 Way Group 0 assigned to Scheme ID 7.	0b0
[27]	S6W3	L2 Cache Way Group 3 mapping to Scheme ID 6. 0b0 Way Group 3 not assigned to Scheme ID 6. 0b1 Way Group 3 assigned to Scheme ID 6.	0b0
[26]	S6W2	L2 Cache Way Group 2 mapping to Scheme ID 6. 0b0 Way Group 2 not assigned to Scheme ID 6. 0b1 Way Group 2 assigned to Scheme ID 6.	0b0
[25]	S6W1	L2 Cache Way Group 1 mapping to Scheme ID 6. 0b0 Way Group 1 not assigned to Scheme ID 6. 0b1 Way Group 1 assigned to Scheme ID 6.	0b0
[24]	S6W0	L2 Cache Way Group 0 mapping to Scheme ID 6. 0b0 Way Group 0 not assigned to Scheme ID 6. 0b1 Way Group 0 assigned to Scheme ID 6.	0b0

Bits	Name	Description	Reset
[23]	S5W3	L2 Cache Way Group 3 mapping to Scheme ID 5. 0b0 Way Group 3 not assigned to Scheme ID 5. 0b1 Way Group 3 assigned to Scheme ID 5.	0b0
[22]	S5W2	L2 Cache Way Group 2 mapping to Scheme ID 5. 0b0 Way Group 2 not assigned to Scheme ID 5. 0b1 Way Group 2 assigned to Scheme ID 5.	0b0
[21]	S5W1	L2 Cache Way Group 1 mapping to Scheme ID 5. 0b0 Way Group 1 not assigned to Scheme ID 5. 0b1 Way Group 1 assigned to Scheme ID 5.	0b0
[20]	S5W0	L2 Cache Way Group 0 mapping to Scheme ID 5. 0b0 Way Group 0 not assigned to Scheme ID 5. 0b1 Way Group 0 assigned to Scheme ID 5.	0b0
[19]	S4W3	L2 Cache Way Group 3 mapping to Scheme ID 4. 0b0 Way Group 3 not assigned to Scheme ID 4. 0b1 Way Group 3 assigned to Scheme ID 4.	0b0
[18]	S4W2	L2 Cache Way Group 2 mapping to Scheme ID 4. 0b0 Way Group 2 not assigned to Scheme ID 4. 0b1 Way Group 2 assigned to Scheme ID 4.	0b0
[17]	S4W1	L2 Cache Way Group 1 mapping to Scheme ID 4. 0b0 Way Group 1 not assigned to Scheme ID 4. 0b1 Way Group 1 assigned to Scheme ID 4.	0b0
[16]	S4W0	L2 Cache Way Group 0 mapping to Scheme ID 4. 0b0 Way Group 0 not assigned to Scheme ID 4. 0b1 Way Group 0 assigned to Scheme ID 4.	0b0

Bits	Name	Description	Reset
[15]	S3W3	L2 Cache Way Group 3 mapping to Scheme ID 3. 0b0 Way Group 3 not assigned to Scheme ID 3. 0b1 Way Group 3 assigned to Scheme ID 3.	0b0
[14]	S3W2	L2 Cache Way Group 2 mapping to Scheme ID 3. 0b0 Way Group 2 not assigned to Scheme ID 3. 0b1 Way Group 2 assigned to Scheme ID 3.	0b0
[13]	S3W1	L2 Cache Way Group 1 mapping to Scheme ID 3. 0b0 Way Group 1 not assigned to Scheme ID 3. 0b1 Way Group 1 assigned to Scheme ID 3.	0b0
[12]	S3W0	L2 Cache Way Group 0 mapping to Scheme ID 3. 0b0 Way Group 0 not assigned to Scheme ID 3. 0b1 Way Group 0 assigned to Scheme ID 3.	0b0
[11]	S2W3	L2 Cache Way Group 3 mapping to Scheme ID 2. 0b0 Way Group 3 not assigned to Scheme ID 2. 0b1 Way Group 3 assigned to Scheme ID 2.	0b0
[10]	S2W2	L2 Cache Way Group 2 mapping to Scheme ID 2. 0b0 Way Group 2 not assigned to Scheme ID 2. 0b1 Way Group 2 assigned to Scheme ID 2.	0b0
[9]	S2W1	L2 Cache Way Group 1 mapping to Scheme ID 2. 0b0 Way Group 1 not assigned to Scheme ID 2. 0b1 Way Group 1 assigned to Scheme ID 2.	0b0
[8]	S2W0	L2 Cache Way Group 0 mapping to Scheme ID 2. 0b0 Way Group 0 not assigned to Scheme ID 2. 0b1 Way Group 0 assigned to Scheme ID 2.	0b0

Bits	Name	Description	Reset
[7]	S1W3	L2 Cache Way Group 3 mapping to Scheme ID 1. 0b0 Way Group 3 not assigned to Scheme ID 1. 0b1 Way Group 3 assigned to Scheme ID 1.	0b0
[6]	S1W2	L2 Cache Way Group 2 mapping to Scheme ID 1. 0b0 Way Group 2 not assigned to Scheme ID 1. 0b1 Way Group 2 assigned to Scheme ID 1.	0b0
[5]	S1W1	L2 Cache Way Group 1 mapping to Scheme ID 1. 0b0 Way Group 1 not assigned to Scheme ID 1. 0b1 Way Group 1 assigned to Scheme ID 1.	0b0
[4]	S1W0	L2 Cache Way Group 0 mapping to Scheme ID 1. 0b0 Way Group 0 not assigned to Scheme ID 1. 0b1 Way Group 0 assigned to Scheme ID 1.	0b0
[3]	S0W3	L2 Cache Way Group 3 mapping to Scheme ID 0. 0b0 Way Group 3 not assigned to Scheme ID 0. 0b1 Way Group 3 assigned to Scheme ID 0.	0b0
[2]	S0W2	L2 Cache Way Group 2 mapping to Scheme ID 0. 0b0 Way Group 2 not assigned to Scheme ID 0. 0b1 Way Group 2 assigned to Scheme ID 0.	0b0
[1]	S0W1	L2 Cache Way Group 1 mapping to Scheme ID 0. 0b0 Way Group 1 not assigned to Scheme ID 0. 0b1 Way Group 1 assigned to Scheme ID 0.	0b0
[0]	S0W0	L2 Cache Way Group 0 mapping to Scheme ID 0. 0b0 Way Group 0 not assigned to Scheme ID 0. 0b1 Way Group 0 assigned to Scheme ID 0.	0b0

Access

MRS <Xt>, IMP_CLUSTERPARTCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b011

MSR IMP_CLUSTERPARTCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b011

Accessibility

MRS <Xt>, IMP_CLUSTERPARTCR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPARTCR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPARTCR_EL1;
```

MSR IMP_CLUSTERPARTCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPORTS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPARTCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPARTCR_EL1 = X[t, 64];
```

A.2.2.57 IMP_CLUSTERQOSR_EL1, Cluster Quality of Service Register

This register controls the Quality of Service aspects for the cluster.

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	0xxx	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-106: AARCH64_IMP_CLUSTERQOSR_EL1 bit assignments

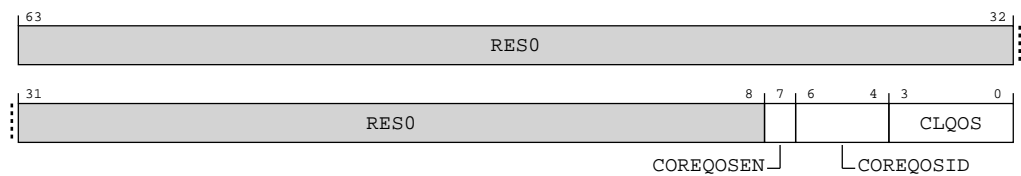


Table A-281: IMP_CLUSTERQOSR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7]	COREQOSEN	Quality of Service enable within the cluster. 0b0 Quality of Service disabled within the cluster. The LCU and the L2 cache arbitrate core requests fairly. 0b1 Quality of Service enabled within the cluster. The LCU and the L2 cache arbitrate requests from core COREQOSID with higher priority than to other cores.	0b0
[6:4]	COREQOSID	Core ID to get higher priority. This field has no effect when COREQOSEN is low.	xxx
[3:0]	CLQOS	Quality of Service setting for the whole cluster. This value will be driven on the Main Manager QoS signals.	0b1110

Access

MRS <Xt>, IMP_CLUSTERQOSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b100

MSR IMP_CLUSTERQOSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b100

Accessibility

MRS <Xt>, IMP_CLUSTERQOSR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERQOSR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERQOSR_EL1;

```

MSR IMP_CLUSTERQOSR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPORTS == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERQOSR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERQOSR_EL1 = X[t, 64];

```

A.2.2.58 IMP_CLUSTERACELSCTLR_EL1, ACELS Port Control Register

This register controls the accesses from the ACELS port.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx11	1111	1111	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

Figure A-107: AARCH64_IMP_CLUSTERACELSTR_EL1 bit assignments

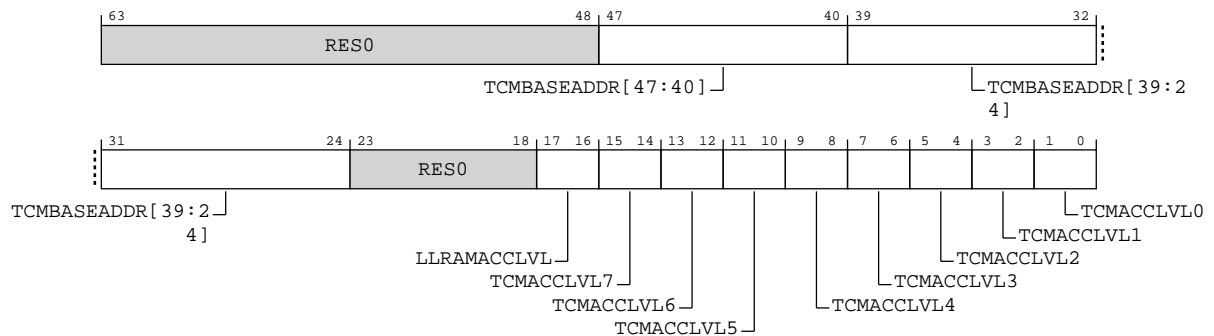


Table A-284: IMP_CLUSTERACELSLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:40]	TCMBASEADDR[47:40]	<p>When PA_W == 48</p> <p>TCMs base address as seen from the ACELS port, bits [47:40]. This field is read-only. This field takes the value of the CFGACELSTCMBASEADDR[47:40] configuration pins.</p> <p>Otherwise</p> <p>RES0</p>	8 {x}
[39:24]	TCMBASEADDR[39:24]	<p>TCMs base address as seen from the ACELS port, bits [39:24] (aligned to 16 MB). This field takes the value of the CFGACELSTCMBASEADDR[39:24] configuration pins.</p> <p>Note:</p> <p>There is only one CFGACELSTCMBASEADDR address for the whole processor. Each core's TCM memory is accessible at an offset from this address:</p> <ul style="list-style-type: none"> Core #0 ITCM: offset 0x0 Core #0 DTCM: offset 0x10_0000 (1 MiB) Core #1 ITCM: offset 0x20_0000 (2 MiB) Core #1 DTCM: offset 0x30_0000 (3 MiB) ... Core #7 ITCM: offset 0xE0_0000 (14 MiB) Core #7 DTCM: offset 0xF0_0000 (15 MiB) <p>Access to this field is: RO</p>	16 {x}
[23:18]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[17:16]	LLRAMACCLVL	LLRAM access level from the ACELS port. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹⁰
[15:14]	TCMACCLVL7	TCM access level from the ACELS port for CPU 7. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹¹
[13:12]	TCMACCLVL6	TCM access level from the ACELS port for CPU 6. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹²
[11:10]	TCMACCLVL5	TCM access level from the ACELS port for CPU 5. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹³

¹⁰ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.¹¹ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.¹² This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.¹³ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.

Bits	Name	Description	Reset
[9:8]	TCMACCLVL4	TCM access level from the ACELS port for CPU 4. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹⁴
[7:6]	TCMACCLVL3	TCM access level from the ACELS port for CPU 3. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹⁵
[5:4]	TCMACCLVL2	TCM access level from the ACELS port for CPU 2. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹⁶
[3:2]	TCMACCLVL1	TCM access level from the ACELS port for CPU 1. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹⁷

¹⁴ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.¹⁵ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.¹⁶ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.¹⁷ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.

Bits	Name	Description	Reset
[1:0]	TCMACCLVLO	TCM access level from the ACELS port for CPU 0. 0b00 No accesses allowed. 0b01 Privileged accesses only allowed. 0b11 Privileged and unprivileged accesses allowed. Other values are Reserved.	0b11 ¹⁸

Access

After a write to AArch64-IMP_CLUSTERACELSTLR_EL1, the programmed permission change will only take effect after there are no new transactions on a given channel. A stream of transactions will use the old permissions until there is an idle cycle and then subsequent transactions will use the new permissions.

MRS <Xt>, IMP_CLUSTERACELSTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b101

MSR IMP_CLUSTERACELSTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b101

Accessibility

After a write to AArch64-IMP_CLUSTERACELSTLR_EL1, the programmed permission change will only take effect after there are no new transactions on a given channel. A stream of transactions will use the old permissions until there is an idle cycle and then subsequent transactions will use the new permissions.

MRS <Xt>, IMP_CLUSTERACELSTLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERACELSTLR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERACELSTLR_EL1;

```

MSR IMP_CLUSTERACELSTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then

```

¹⁸ This field is preserved on a reset when exiting either MEM_RET or MEM_RET_EMU mode.


```
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPORTS == '0' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      IMP_CLUSTERACELSTCLR_EL1 = X[t, 64];
  elseif PSTATE.EL == EL2 then
    IMP_CLUSTERACELSTCLR_EL1 = X[t, 64];
```

A.2.2.59 IMP_CLUSTERMEMPROTCTLR_EL1, Cluster Memory Protection Control Register

This register controls the memory protection features of the cluster.

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-108: AARCH64_IMP_CLUSTERMEMPROTCTLR_EL1 bit assignments

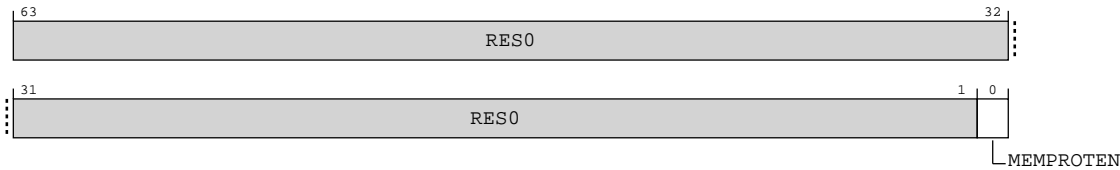


Table A-287: IMP_CLUSTERMEMPROTCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	MEMPROTEN	<p>When RAM_PROTECTION == 1</p> <p>Memory protection enable.</p> <p>0b0</p> <p>Structures required to support internal RAM protection functionality are turned off.</p> <p>0b1</p> <p>Structures required to support internal RAM protection functionality are turned on.</p> <ul style="list-style-type: none"> This field must not be programmed while the caches are enabled, otherwise the behavior of the processor memory system becomes UNPREDICTABLE. Software must ensure that the processor caches are disabled before programming this field. After programming this field, software must ensure that the caches are invalidated before they can be enabled again. If the values of AArch64-IMP_CLUSTERMEMPROTCTLR_EL1.MEMPROTEN and AArch64-IMP_MEMPROTCTLR_EL1.MEMPROTEN for all cores in the processor cluster are not the same, it is possible for wrong data values to be consumed by any core in the cluster, because errors that are locally suppressed by disabling RAM protection may spread through hardware data coherency mechanisms. Arm recommends that AArch64-IMP_CLUSTERMEMPROTCTLR_EL1.MEMPROTEN and all AArch64-IMP_MEMPROTCTLR_EL1.MEMPROTEN fields are programmed with the same value, unless otherwise directed by Arm. <p>Otherwise</p> <p>RES0</p>	<p>When the CFGRAMPROTEN configuration pin is HIGH</p> <p>'1'</p> <p>When the CFGRAMPROTEN configuration pin is LOW</p> <p>'0'</p>

Access

MRS <Xt>, IMP_CLUSTERMEMPROTCTLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b110

MSR IMP_CLUSTERMEMPROTCTLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0100	0b110

Accessibility

MRS <Xt>, IMP_CLUSTERMEMPROTCTLR_EL1

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

```
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      X[t, 64] = IMP_CLUSTERMEMPROTCTLR_EL1;
  elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERMEMPROTCTLR_EL1;
```

MSR IMP_CLUSTERMEMPROTCTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
  UNDEFINED;
elseif PSTATE.EL == EL1 then
  if HCR_EL2.TIDCP == '1' || ACTLR_EL2.DIAGNOSTIC == '0' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    IMP_CLUSTERMEMPROTCTLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
  IMP_CLUSTERMEMPROTCTLR_EL1 = X[t, 64];
```

A.2.2.60 AIDR_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-109: AARCH64_AIDR_EL1 bit assignments

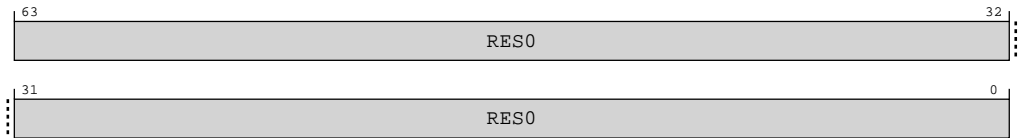


Table A-290: AIDR_EL1 bit descriptions

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

A.2.2.61 IMP_CDBGDR0_EL1, Cache Debug Data Register 0

Contains data from a preceeding cache debug operation.

This register is populated after one of the following operations have been executed:

- SYS IMP_CDBGDCD
- SYS IMP_CDBGDCT
- SYS IMP_CDBGICD
- SYS IMP_CDBGICT
- SYS IMP_CDBGTD
- SYS IMP_CDBGTT

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When the last cache debug operation was SYS IMP_CDBGDCD, or the last cache debug operation was SYS IMP_CDBGICD or the last cache debug operation was SYS IMP_CDBGDTD

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

When the last cache debug operation was SYS IMP_CDBGDCT

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

When the last cache debug operation was SYS IMP_CDBGICT

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

When the last cache debug operation was SYS IMP_CDBGTD and index is in range 0x000 - 0xFF

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

When the last cache debug operation was SYS IMP_CDBGTD and index is in range 0x100 - 0x107

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

When the last cache debug operation was SYS IMP_CDBGTT and index is in range 0x000 - 0xFF

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

When the last cache debug operation was SYS IMP_CDBGTT and index is in range 0x100 - 0x107

XXXX XXXX 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 the last cache debug operation was SYS IMP_CDBGDCD, or the last cache debug operation was SYS IMP_CDBGICD or the last cache debug operation was SYS IMP_CBGDTD

Figure A-110: AARCH64_IMP_CDBGDR0_EL1 bit assignments

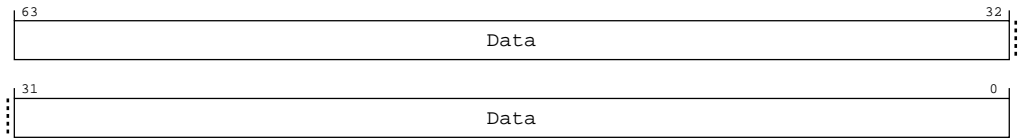


Table A-292: IMP_CDBGDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Data	Data contents of cache at specified Set/Way/Offset	64 { x }

When the last cache debug operation was SYS IMP_CDBGDCT

Figure A-111: AARCH64_IMP_CDBGDR0_EL1 bit assignments

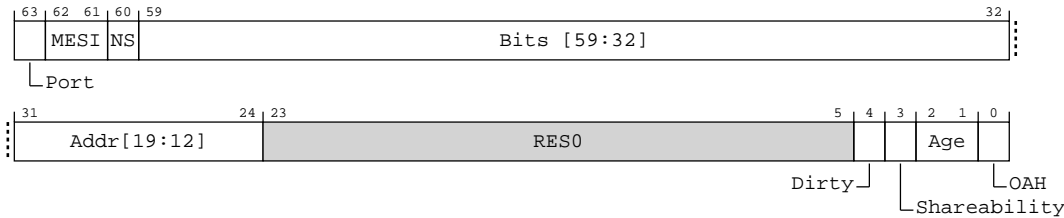


Table A-293: IMP_CDBGDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	Port	Port (from Tag contents, not stored in RAM) 0b0 Main Manager port 0b1 LLRAM port	x
[62:61]	MESI	MESI State (from Tag RAM) 0b00 Invalid 0b01 Shared 0b10 Unique 0b11 Unique (Transient)	xx
[60]	NS	Non-secure state (from Tag RAM)	x

Bits	Name	Description	Reset
[59:32]	Addr[47:20]	When PA_W == 48 Tag Address, Physical Address [47:20] (from Tag RAM). Otherwise Tag Address, Physical Address [39:12] (from Tag RAM).	28 {x}
[31:24]	Addr[19:12]	When PA_W == 48 Tag Address, Physical Address [19:12] (from Tag RAM). Otherwise RES0	8 {x}
[23:5]	RES0	Reserved	RES0
[4]	Dirty	Dirty bit (from Dirty RAM) 0b0 Clean 0b1 Modified/Dirty	x
[3]	Shareability	Shareability (from Dirty RAM)	x
[2:1]	Age	Age (from Dirty RAM)	xx
[0]	OAH	Outer Allocation Hint (from Dirty RAM)	x

When the last cache debug operation was SYS IMP_CDBGICT

Figure A-112: AARCH64_IMP_CDBGDR0_EL1 bit assignments

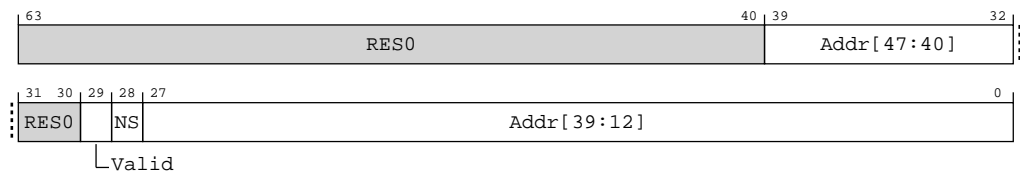
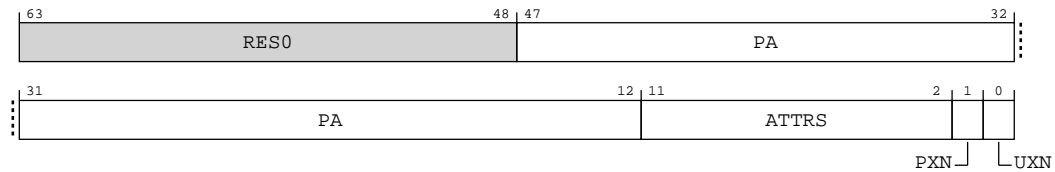


Table A-294: IMP_CDBGDR0_EL1 bit descriptions

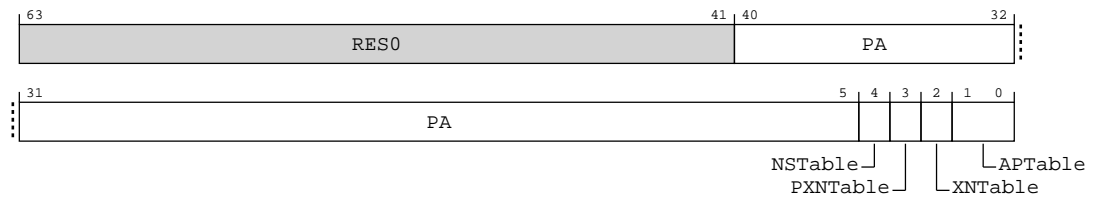
Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39:32]	Addr[47:40]	When PA_W == 48 Tag Address, Physical Address [47:40] (from Tag RAM). Otherwise RES0	8 {x}
[31:30]	RES0	Reserved	RES0
[29]	Valid	Valid	x
[28]	NS	Non-secure state	x
[27:0]	Addr[39:12]	Tag Address, Physical Address [39:12]	28 {x}

When the last cache debug operation was SYS IMP_CDBGTD and index is in range 0x0000x0FF

Figure A-113: AARCH64_IMP_CDBGDR0_EL1 bit assignments**Table A-295: IMP_CDBGDR0_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:12]	PA	The Physical Address.	36{x}
[11:2]	ATTRS	Defines the memory attribute.	10{x}
[1]	PXN	Executable in stage 1 non-user mode.	x
[0]	UXN	Executable in stage 1 user mode.	x

When the last cache debug operation was SYS IMP_CDBGTD and index is in range 0x1000x107

Figure A-114: AARCH64_IMP_CDBGDR0_EL1 bit assignments**Table A-296: IMP_CDBGDR0_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:41]	RES0	Reserved	RES0
[40:5]	PA	The physical base address of L3 descriptor. Note: This is the actual physical address, but a stage 2 lookup will still be required to obtain the combined attributes.	36{x}
[4]	NSTable	Combined NSTable bits from stage 1 descriptors up to last level.	x
[3]	PXNTable	Stores the stage 1 privilege execution permission information up to last level, starts from executable (0b0) and combined with PXNTable bits from stage 1 descriptors up to last level.	x
[2]	XNTable	Stores the stage 1 execution permission information up to last level, starts from executable (0b0) and combined with XNTable bits from stage 1 descriptors up to last level.	x
[1:0]	APTable	Stores the stage 1 access permission information up to last level, starts from full access (0b01) and combined with APTable bits from stage 1 descriptors up to last level.	xx

When the last cache debug operation was SYS IMP_CDBGTT and index is in range 0x0000x0FF

Figure A-115: AARCH64_IMP_CDBGDR0_EL1 bit assignments

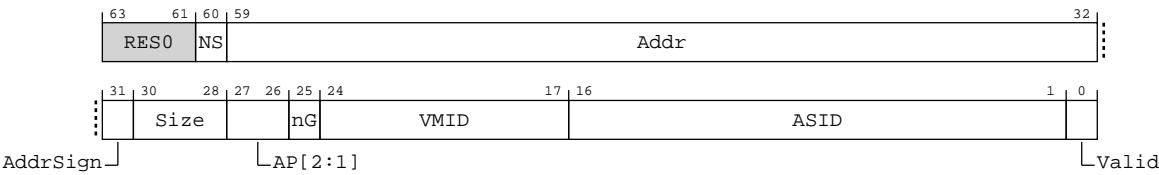


Table A-297: IMP_CDBGDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:61]	RES0	Reserved	RES0
[60]	NS	The security state allocated to this memory region.	x
[59:32]	Addr	Virtual Address. Depending on the page size, unused lower bits are 0.	28 {x}
[31]	AddrSign	The Virtual Address Sign bit, VA[48].	x
[30:28]	Size	Page or block size of the stage 1 translation result.	xxx
[27:26]	AP[2:1]	Data Access Permissions bits.	xx
[25]	nG	The not global bit.	x
[24:17]	VMID	Virtual machine identifier.	8 {x}
[16:1]	ASID	Address space identifier. This field will be 0 if ASID is not used.	16 {x}
[0]	Valid	Valid	x

When the last cache debug operation was SYS IMP_CDBGTT and index is in range 0x1000x107

Figure A-116: AARCH64_IMP_CDBGDR0_EL1 bit assignments

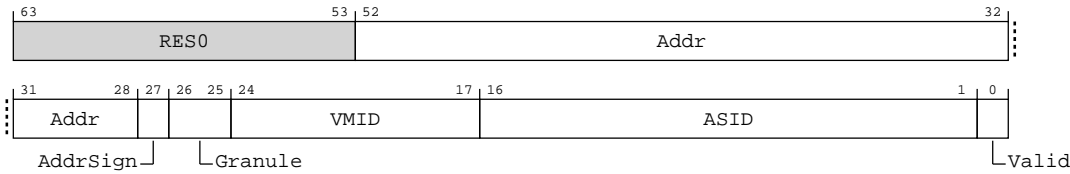


Table A-298: IMP_CDBGDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:53]	RES0	Reserved	RES0
[52:28]	Addr	Virtual Address. Depending on the page size, unused lower bits are 0.	25 {x}
[27]	AddrSign	The Virtual Address Sign bit, VA[48].	x
[26:25]	Granule	Translation granule size.	xx
[24:17]	VMID	Virtual machine identifier.	8 {x}
[16:1]	ASID	Address space identifier.	16 {x}
[0]	Valid	Valid	x

Access

MRS <Xt>, IMP_CDBGDR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b0000	0b000

Accessibility

MRS <Xt>, IMP_CDBGDR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CDBGDR0_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CDBGDR0_EL1;
```

A.2.2.62 IMP_CDBGDR1_EL1, Cache Debug Data Register 1

Contains data from a preceeding cache debug operation.

This register is populated or zeroed after one of the following operations have been executed:

- SYS IMP_CDBGDCD
- SYS IMP_CDBGDCT
- SYS IMP_CDBGICD
- SYS IMP_CDBGICT
- SYS IMP_CDBGTD
- SYS IMP_CDBGTT

Configurations

This register is available in all configurations.

Attributes

Width

64

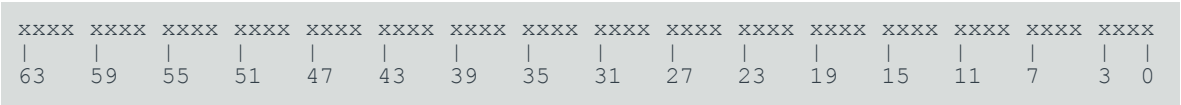
Functional group

Generic System Control

Access type

See bit descriptions

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-117: AARCH64_IMP_CDBGDR1_EL1 bit assignments

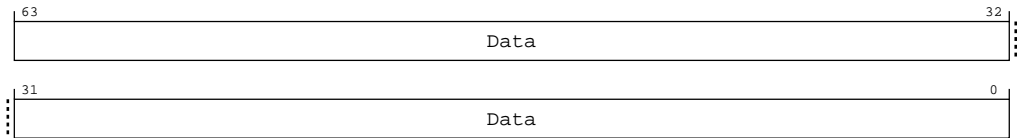


Table A-300: IMP_CDBGDR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Data	Data contents of cache at specified Set/Way/Offset	64 { x }

Access

MRS <Xt>, IMP_CDBGDR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b0000	0b001

Accessibility

MRS <Xt>, IMP_CDBGDR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CDBGDR1_EL1;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CDBGDR1_EL1;
```

A.2.2.63 IMP_CLUSTERCDBGDR0_EL1, Cluster Cache Debug Data Register 0

Contains data from a preceeding cache debug operation.

This register is populated after one of the following operations have been executed:

- SYS IMP_CLUSTERCDBGL2D
- SYS IMP_CLUSTERCDBGL2T
- SYS IMP_CLUSTERCDBGL2DT

- SYS IMP_CLUSTERCDBGLCUdT

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

When the last cache debug operation was SYS IMP_CLUSTERCDBGL2D

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

When the last cache debug operation was SYS IMP_CLUSTERCDBGL2T and PA_W == 40

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

When the last cache debug operation was SYS IMP_CLUSTERCDBGL2T and PA_W == 48

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

When the last cache debug operation was SYS IMP_CLUSTERCDBGL2DT and PA_W == 40

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

When the last cache debug operation was SYS IMP_CLUSTERCDBGL2DT and PA_W == 48

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

When the last cache debug operation was SYS IMP_CLUSTERCDBGLCUdT

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 the last cache debug operation was SYS IMP_CLUSTERCDBGL2D

Figure A-118: AARCH64_IMP_CLUSTERCDBGDR0_EL1 bit assignments

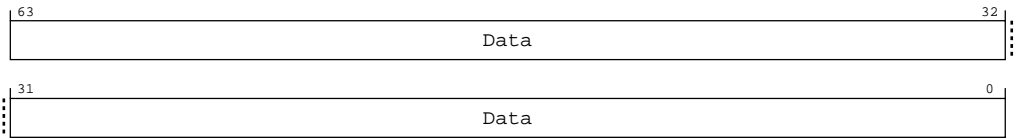
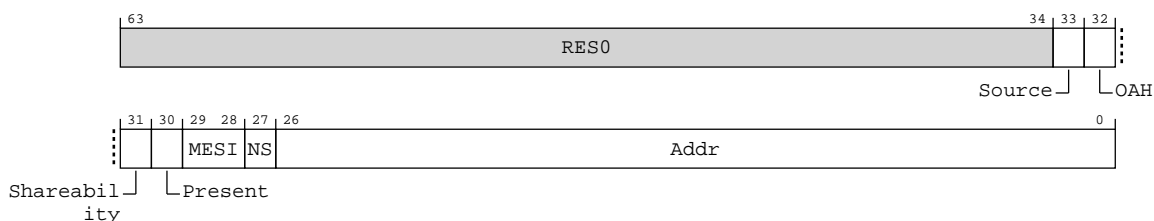


Table A-302: IMP_CLUSTERCDBGDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Data	Data contents of cache at specified Set/Way/Offset	64 {x}

When the last cache debug operation was SYS IMP_CLUSTERCDBGDL2T and PA_W == 40

Figure A-119: AARCH64_IMP_CLUSTERCDBGDR0_EL1 bit assignments**Table A-303: IMP_CLUSTERCDBGDR0_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:34]	RES0	Reserved	RES0
[33]	Source	Cache line source	x
[32]	OAH	Outer Allocation Hint (from Dirty RAM)	x
[31]	Shareability	Shareability (from Dirty RAM)	x
[30]	Present	Cache line is present in the L1 cache of any of the cores in this cluster.	x
[29:28]	MESI	MESI State (from Tag RAM) 0b00 Invalid 0b01 Shared Clean 0b10 Unique Dirty 0b11 Unique Clean	xx
[27]	NS	Non-secure state (from Tag RAM)	x
[26:0]	Addr	Tag Address, Physical Address [39:12] (from Tag RAM)	27 {x}

When the last cache debug operation was SYS IMP_CLUSTERCDBGDL2T and PA_W == 48

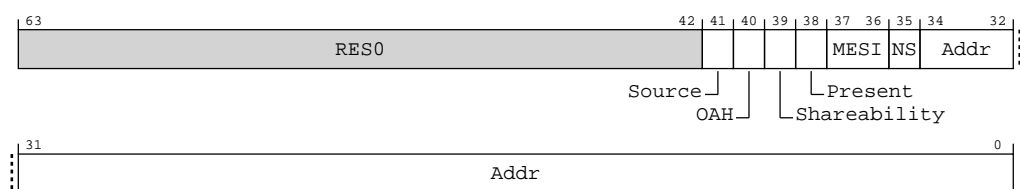
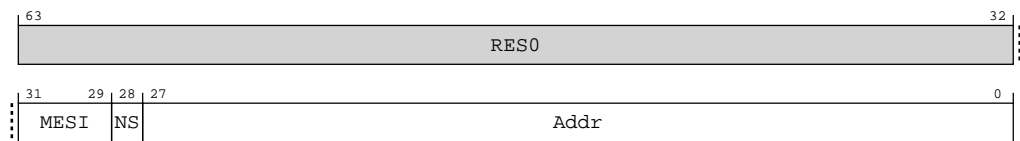
Figure A-120: AARCH64_IMP_CLUSTERCDBGDR0_EL1 bit assignments

Table A-304: IMP_CLUSTERCDBGDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:42]	RES0	Reserved	RES0
[41]	Source	Cache line source	x
[40]	OAH	Outer Allocation Hint (from Dirty RAM)	x
[39]	Shareability	Shareability (from Dirty RAM)	x
[38]	Present	Cache line is present in the L1 cache of any of the cores in this cluster.	x
[37:36]	MESI	MESI State (from Tag RAM) 0b00 Invalid 0b01 Shared Clean 0b10 Unique Dirty 0b11 Unique Clean	xx
[35]	NS	Non-secure state (from Tag RAM)	x
[34:0]	Addr	Tag Address, Physical Address [47:12] (from Tag RAM)	35 {x}

When the last cache debug operation was SYS IMP_CLUSTERCDBGDL2DT and PA_W == 40

Figure A-121: AARCH64_IMP_CLUSTERCDBGDR0_EL1 bit assignments**Table A-305: IMP_CLUSTERCDBGDR0_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:29]	MESI	MESI State (from Tag RAM) 0b000 Invalid 0b001 Cluster unique, CPU shared clean, non-shareable 0b010 Cluster unique, CPU shared dirty, non-shareable 0b011 Cluster unique, CPU unique, non-shareable 0b100 Cluster unique, CPU shared clean, shareable 0b101 Cluster unique, CPU shared dirty, shareable 0b110 Cluster unique, CPU unique, shareable 0b111 Cluster shared, CPU shared, shareable	xxx
[28]	NS	Non-secure state (from Tag RAM)	x
[27:0]	Addr	Tag Address, Physical Address [39:12] (from Tag RAM)	28 {x}

When the last cache debug operation was SYS IMP_CLUSTERCDBGL2DT and PA_W == 48

Figure A-122: AARCH64_IMP_CLUSTERCDBGDR0_EL1 bit assignments

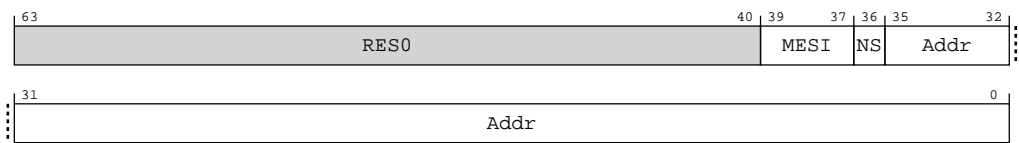


Table A-306: IMP_CLUSTERCDBGDR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[39:37]	MESI	MESI State (from Tag RAM) 0b000 Invalid 0b001 Cluster unique, CPU shared clean, non-shareable 0b010 Cluster unique, CPU shared dirty, non-shareable 0b011 Cluster unique, CPU unique, non-shareable 0b100 Cluster unique, CPU shared clean, shareable 0b101 Cluster unique, CPU shared dirty, shareable 0b110 Cluster unique, CPU unique, shareable 0b111 Cluster shared, CPU shared, shareable	xxx
[36]	NS	Non-secure state (from Tag RAM)	x
[35:0]	Addr	Tag Address, Physical Address [47:12] (from Tag RAM)	36 {x}

When the last cache debug operation was SYS IMP_CLUSTERCDBGDGLCUDT

Figure A-123: AARCH64_IMP_CLUSTERCDBGDRO_EL1 bit assignments

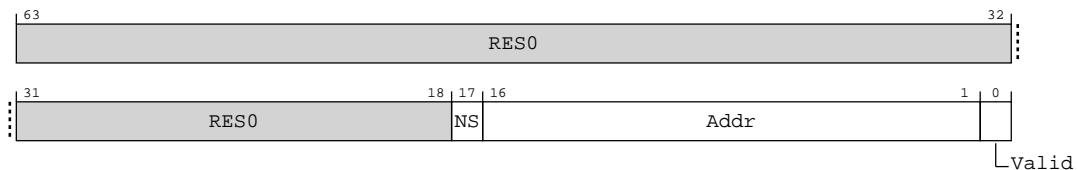


Table A-307: IMP_CLUSTERCDBGDRO_EL1 bit descriptions

Bits	Name	Description	Reset
[63:18]	RES0	Reserved	RES0
[17]	NS	Non-secure state (from Tag RAM)	x
[16:1]	Addr	Tag Address (from Tag RAM) The tag width depends on the implemented cache size. The unused bits are RES0 . The LLRAM base address is excluded from the stored address bits.	16 {x}
[0]	Valid	Valid	x

Access

MRS <Xt>, IMP_CLUSTERCDBGDRO_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b0011	0b000

Accessibility

MRS <Xt>, IMP_CLUSTERCDBGDRO_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERCDBGDRO_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERCDBGDRO_EL1;

```

A.2.2.64 IMP_CPUPSELR_EL1, Instruction Patch Selection Register

Selects the current instruction patch register <n> for subsequent accesses to AArch64-IMP_CPUPCR<n>_EL1, AArch64-IMP_CPUPOR<n>_EL1, and AArch64-IMP_CPUPMR<n>_EL1.

It is used in conjunction with AArch64-IMP_CPUPCR_EL1, AArch64-IMP_CPUPOR_EL1, and AArch64-IMP_CPUPMR_EL1 respectively, to access the selected register.

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-124: AARCH64_IMP_CPUPSELR_EL1 bit assignments

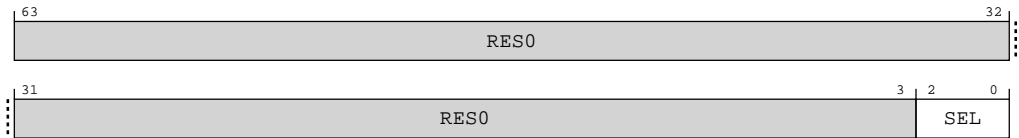


Table A-309: IMP_CPUPSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2:0]	SEL	Selects the current instruction patch register <n>. If the value is greater than or equal to the number of accessible patch registers, subsequent reads and writes of AArch64-IMP_CPUPCR_EL1, AArch64-IMP_CPUPOR_EL1, and AArch64-IMP_CPUPMR_EL1 behave as RAZ/WI .	xxx

Access

MRS <Xt>, IMP_CPUPSELR_EL1

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

MSR IMP_CPUPSELR_EL1, <Xt>

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

Accessibility

MRS <Xt>, IMP_CPUPSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPSELR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPSELR_EL1;
```

MSR IMP_CPUPSELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPSELR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
```

```
IMP_CPUPSELR_EL1 = X[t, 64];
```

A.2.2.65 IMP_CPUPCR_EL1, Selected Instruction Patch Control Register

This register accesses one of the AArch64-IMP_CPUPCR<n>_EL1 registers, selected by AArch64-IMP_CPUPSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

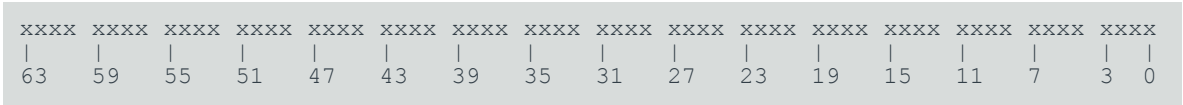
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-125: AARCH64_IMP_CPUPCR_EL1 bit assignments

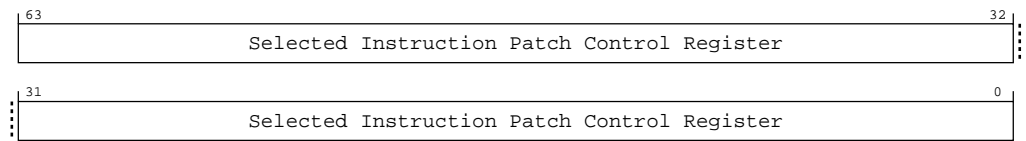


Table A-312: IMP_CPUPCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	This register accesses AArch64-IMP_CPUPCR<n>_EL1 where n is the value in AArch64-IMP_CPUPSELR_EL1.SEL.	64 {x}

Access

If AArch64-IMP_CPUPSEL_{EL1} is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPCR_{EL1} behave as **RAZ/WI**.

MRS <Xt>, IMP_CPUPCR_{EL1}

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b001

MSR IMP_CPUPCR_{EL1}, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b001

Accessibility

If AArch64-IMP_CPUPSEL_{EL1} is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPCR_{EL1} behave as RAZ/WI.

MRS <Xt>, IMP_CPUPCR_{EL1}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPCR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPCR_EL1;

```

MSR IMP_CPUPCR_{EL1}, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CPUPCR_EL1 = X[t, 64];

```

A.2.2.66 IMP_CPUPCR<n>_EL1, Instruction Patch Control Registers, n = 0 - 5

Configures Instruction Patch n, where n is 0 to 5. This register provides **IMPLEMENTATION DEFINED** configuration and control options for the core.

Configurations

This register is available in all configurations.

Attributes

Width

64

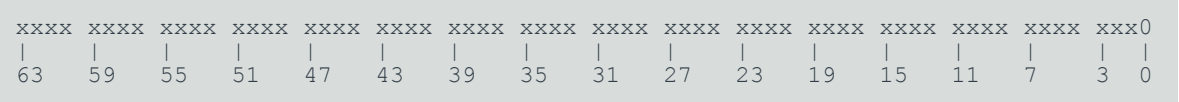
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-126: AARCH64_IMP_CPUPCR<n>_EL1 bit assignments

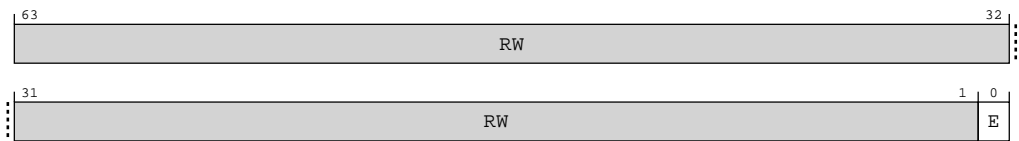


Table A-315: IMP_CPUPCR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RW	Reserved	RW
[0]	E	Enable. 0b0 Any core control options affected by IMP_CPUPCR<n>_EL1, AArch64-IMP_CPUPMR<n>_EL1 and AArch64-IMP_CPUPOR<n>_EL1 are disabled. 0b1 IMPLEMENTATION DEFINED core control options enabled. Arm strongly recommends that you do not set this field to 0b1 unless directed by Arm.	0b0

Access

Writing to this register might cause **UNPREDICTABLE** behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

IMP_CPUPCR<n>_EL1 can only be accessed by using AArch64-IMP_CPUPCR_EL1 with AArch64-IMP_CPUPSELR_EL1.SEL set to n.

If <n> is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPCR<n>_EL1 behave as **RAZ/WI**.

MRS <Xt>, IMP_CPUPCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b001

MSR IMP_CPUPCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b001

Accessibility

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

IMP_CPUPCR<n>_EL1 can only be accessed by using AArch64-IMP_CPUPCR_EL1 with AArch64-IMP_CPUPSELR_EL1.SEL set to n.

If <n> is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPCR<n>_EL1 behave as RAZ/WI.

MRS <Xt>, IMP_CPUPCR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPCR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPCR_EL1;
```

MSR IMP_CPUPCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CPUPCR_EL1 = X[t, 64];
```

A.2.2.67 IMP_CPUPOR_EL1, Selected Instruction Patch Opcode Register

This register accesses one of the AArch64-IMP_CPUPOR<n>_EL1 registers, selected by AArch64-IMP_CPUPSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

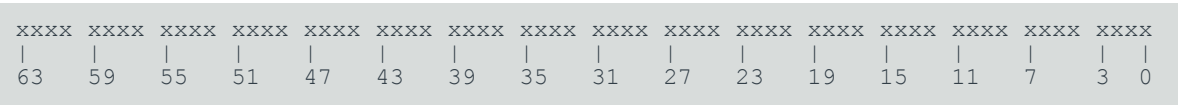
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-127: AARCH64_IMP_CPUPOR_EL1 bit assignments

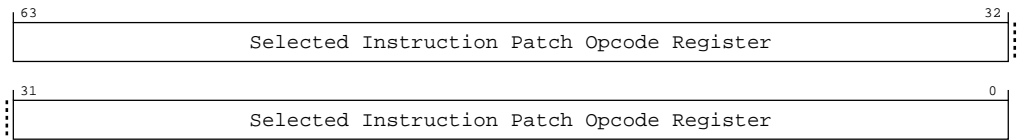


Table A-318: IMP_CPUPOR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	This register accesses AArch64-IMP_CPUPOR<n>_EL1 where n is the value in AArch64-IMP_CPUPSELR_EL1.SEL.	64 { x }

Access

If AArch64-IMP_CPUPSELR_EL1 is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPOR_EL1 behave as **RAZ/WI**.

MRS <Xt>, IMP_CPUPOR_EL1

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

MSR IMP_CPUPOR_EL1, <Xt>

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

Accessibility

If AArch64-IMP_CPUPSELR_EL1 is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPOR_EL1 behave as RAZ/WI.
MRS <Xt>, IMP_CPUPOR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPOR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPOR_EL1;
```

MSR IMP_CPUPOR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPOR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CPUPOR_EL1 = X[t, 64];
```

A.2.2.68 IMP_CPUPOR<n>_EL1, Instruction Patch Opcode Registers, n = 0 - 5

Opcode for Instruction Patch n, where n is 0 to 5. This register provides **IMPLEMENTATION DEFINED** configuration and control options for the core. This register has no effect when AArch64-IMP_CPUPCR<n>_EL1.E is set to 0b0.

Configurations

This register is available in all configurations.

Attributes

Width

64

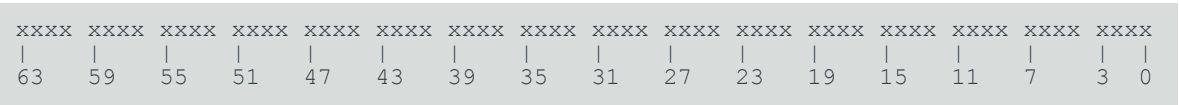
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-128: AARCH64_IMP_CPUPOR<n>_EL1 bit assignments

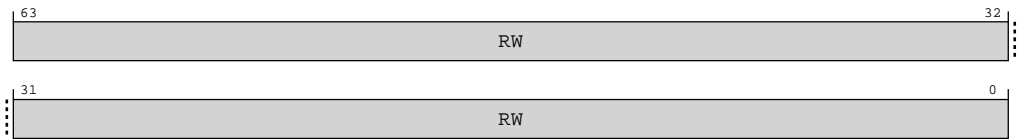


Table A-321: IMP_CPUPOR<n>_EL1 bit descriptions

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

Access

Writing to this register might cause **UNPREDICTABLE** behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

IMP_CPUPOR<n>_EL1 can only be accessed by using AArch64-IMP_CPUPOR_EL1 with AArch64-IMP_CPUPSELR_EL1.SEL set to n.

If <n> is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPOR<n>_EL1 behave as **RAZ/WI**.

MRS <Xt>, IMP_CPUPOR_EL1

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

MSR IMP_CPUPOR_EL1, <Xt>

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

Accessibility

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

IMP_CPUPOR<n>_EL1 can only be accessed by using AArch64-IMP_CPUPOR_EL1 with AArch64-IMP_CPUPSELR_EL1.SEL set to n.

If <n> is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPOR<n>_EL1 behave as RAZ/WI.

MRS <Xt>, IMP_CPUPOR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CPUPOR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPOR_EL1;

```

MSR IMP_CPUPOR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPOR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CPUPOR_EL1 = X[t, 64];

```

A.2.2.69 IMP_CPUPMR_EL1, Selected Instruction Patch Mask Register

This register accesses one of the AArch64-IMP_CPUPMR<n>_EL1 registers, selected by AArch64-IMP_CPUPSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

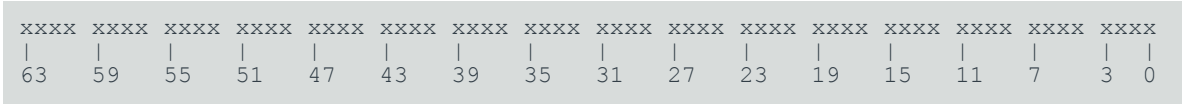
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-129: AARCH64_IMP_CPUPMR_EL1 bit assignments

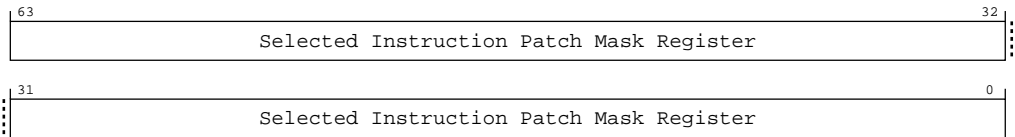


Table A-324: IMP_CPUPMR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	This register accesses AArch64-IMP_CPUPMR<n>_EL1 where n is the value in AArch64-IMP_CPUPSELR_EL1.SEL.	64 {x}

Access

If AArch64-IMP_CPUPSELR_EL1 is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPMR_EL1 behave as **RAZ/WI**.

MRS <Xt>, IMP_CPUPMR_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b011

MSR IMP_CPUPMR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b011

Accessibility

If AArch64-IMP_CPUPSELR_EL1 is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPMR_EL1 behave as RAZ/WI.

MRS <Xt>, IMP_CPUPMR_EL1

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

```
if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = IMP_CPUPMR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPMR_EL1;
```

MSR IMP_CPUPMR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPMR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CPUPMR_EL1 = X[t, 64];
```

A.2.2.70 IMP_CPUPMR<n>_EL1, Instruction Patch Mask Registers, n = 0 - 5

Mask for Instruction Patch n, where n is 0 to 5. This register provides **IMPLEMENTATION DEFINED** configuration and control options for the core. This register has no effect when AArch64-IMP_CPUPCR<n>_EL1.E is set to 0b0.

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-130: AARCH64_IMP_CPUPMR<n>_EL1 bit assignments

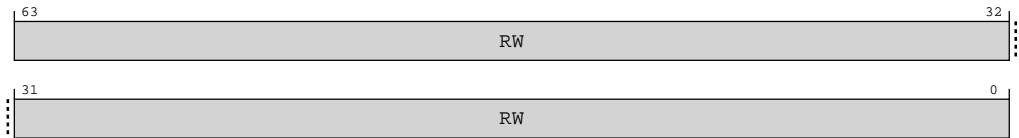


Table A-327: IMP_CPUPMR<n>_EL1 bit descriptions

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

Access

Writing to this register might cause **UNPREDICTABLE** behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

IMP_CPUPMR<n>_EL1 can only be accessed by using AArch64-IMP_CPUPMR_EL1 with AArch64-IMP_CPUPSELR_EL1.SEL set to n.

If <n> is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPMR<n>_EL1 behave as **RAZ/WI**.

MRS <Xt>, IMP_CPUPMR_EL1

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b011

MSR IMP_CPUPMR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b010	0b1111	0b1000	0b011

Accessibility

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

IMP_CPUPMR<n>_EL1 can only be accessed by using AArch64-IMP_CPUPMR_EL1 with AArch64-IMP_CPUPSELR_EL1.SEL set to n.

If <n> is greater than or equal to the number of accessible patch registers, reads and writes of IMP_CPUPMR<n>_EL1 behave as RAZ/WI.

MRS <Xt>, IMP_CPUPMR_EL1

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

```
if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = IMP_CPUPMR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CPUPMR_EL1;
```

MSR IMP_CPUPMR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.PATCH == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CPUPMR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CPUPMR_EL1 = X[t, 64];
```

A.2.2.71 NZCV, Condition Flags

Allows access to the condition flags.

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-131: AARCH64_NZCV bit assignments

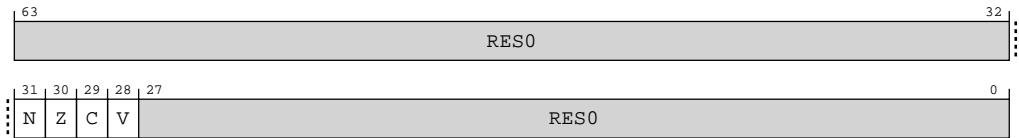


Table A-330: NZCV bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	N	Negative condition flag. Set to 1 if the result of the last flag-setting instruction was negative.	x
[30]	Z	Zero condition flag. Set to 1 if the result of the last flag-setting instruction was zero, and to 0 otherwise. A result of zero often indicates an equal result from a comparison.	x
[29]	C	Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.	x
[28]	V	Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.	x
[27:0]	RES0	Reserved	RES0

Access

MRS <Xt>, NZCV

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b000

MSR NZCV, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b000

Accessibility

MRS <Xt>, NZCV

```
if PSTATE.EL == EL0 then
    X[t, 64] = Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
elseif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
elseif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
```

MSR NZCV, <Xt>

```
if PSTATE.EL == EL0 then
    PSTATE.<N,Z,C,V> = X[t, 64]<31:28>;
elseif PSTATE.EL == EL1 then
    PSTATE.<N,Z,C,V> = X[t, 64]<31:28>;
```

```
elseif PSTATE.EL == EL2 then
    PSTATE.<N,Z,C,V> = X[t, 64]<31:28>;
```

A.2.2.72 DAIF, Interrupt Mask Bits

Allows access to the interrupt mask bits.

Configurations

This register is available in all configurations.

Attributes

Width

64

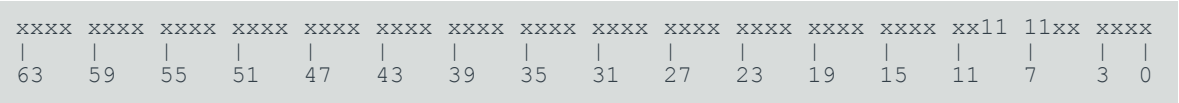
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-132: AARCH64_DAIF bit assignments

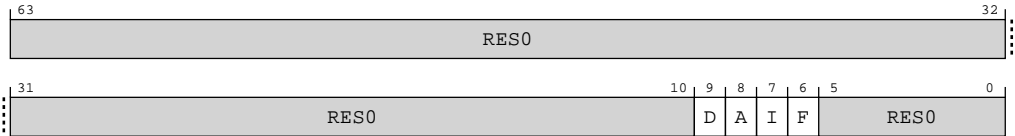


Table A-333: DAIF bit descriptions

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

Bits	Name	Description	Reset
[9]	D	Process state D mask. 0b0 Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are not masked. 0b1 Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are masked. When the target Exception level of the debug exception is higher than the current Exception level, the exception is not masked by this bit.	0b1
[8]	A	SError interrupt mask bit. 0b0 Exception not masked. 0b1 Exception masked.	0b1
[7]	I	IRQ mask bit. 0b0 Exception not masked. 0b1 Exception masked.	0b1
[6]	F	FIQ mask bit. 0b0 Exception not masked. 0b1 Exception masked.	0b1
[5:0]	RES0	Reserved	RES0

Access

MRS <Xt>, DAIF

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b001

MSR DAIF, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b001

MSR DAIFSet, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b110

MSR DAIFClr, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b111

Accessibility

MRS <Xt>, DAIF

```

if PSTATE.EL == EL0 then
    if SCTLRL1.UMA == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            X[t, 64] = Zeros(54):PSTATE.<D,A,I,F>:Zeros(6);
    elseif PSTATE.EL == EL1 then
        X[t, 64] = Zeros(54):PSTATE.<D,A,I,F>:Zeros(6);
    elseif PSTATE.EL == EL2 then
        X[t, 64] = Zeros(54):PSTATE.<D,A,I,F>:Zeros(6);

```

MSR DAIF, <Xt>

```

if PSTATE.EL == EL0 then
    if SCTLRL1.UMA == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            PSTATE.<D,A,I,F> = X[t, 64]<9:6>;
    elseif PSTATE.EL == EL1 then
        PSTATE.<D,A,I,F> = X[t, 64]<9:6>;
    elseif PSTATE.EL == EL2 then
        PSTATE.<D,A,I,F> = X[t, 64]<9:6>;

```

MSR DAIFSet, #<imm>

MSR DAIFClr, #<imm>

A.2.2.73 DIT, Data Independent Timing

Allows access to the Data Independent Timing bit.

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	xxx0	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-133: AARCH64_DIT bit assignments

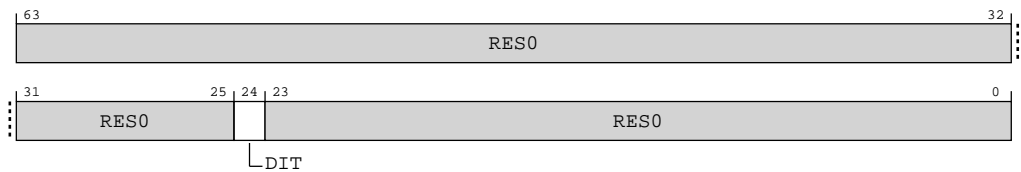


Table A-338: DIT bit descriptions

Bits	Name	Description	Reset
[63:25]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[24]	DIT	<p>Data Independent Timing.</p> <p>0b0</p> <p>The architecture makes no statement about the timing properties of any instructions.</p> <p>0b1</p> <p>The architecture requires that:</p> <ul style="list-style-type: none"> • The timing of every load and store instruction is insensitive to the value of the data being loaded or stored. • For certain data processing instructions, the instruction takes a time which is independent of: <ul style="list-style-type: none"> ◦ The values of the data supplied in any of its registers. ◦ The values of the NZCV flags. • For certain data processing instructions, the response of the instruction to asynchronous exceptions does not vary based on: <ul style="list-style-type: none"> ◦ The values of the data supplied in any of its registers. ◦ The values of the NZCV flags. <p>The data processing instructions affected by this bit are:</p> <ul style="list-style-type: none"> • A subset of those instructions which use the general-purpose register file. These instructions are: <ul style="list-style-type: none"> ◦ ADC, ADCS, ADD, ADDS, AND, ANDS, ASR, ASRV, BFC, BFI, BFM, BFXIL, BIC, BICS, CCMN, CCMP, CFINV, CINC, CINV, CLS, CLZ, CMN, CMP, CNEG, CSEL, CSET, CSETM, CSINC, CSINV, CSNEG, EON, EOR, EXTR, LSL, LSLV, LSR, LSRV, MADD, MNEG, MOV, MOVK, MVN, MOVZ, MSUB, MUL, MVN, NEG, NEGS, NGC, NGCS, NOP, ORN, ORR, RBIT, REV, REV16, REV32, REV64, RMIF, ROR, RORV, SBC, SBSCS, SBFIZ, SBFM, SBFX, SETF8, SETF16, SMADDL, SMNEGL, SMSUBL, SMULH, SMULL, SUB, SUBS, SXTB, SXTH, SXTW, TST, UBFIZ, UBFM, UBFX, UMADDL, UMNEGL, UMSUBL, UMULH, UMULL, UXTB, and UXTH. ◦ CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX. • A subset of those instructions which use the SIMD&FP register file. These instructions are: <ul style="list-style-type: none"> ◦ ABS, ADD, ADDHN, ADDHN2, ADDP, ADDV, AND, BIC, BIF, BIT, BSL, CLS, CLZ, CMEQ, CMGE, CMGT, CMHI, CMHS, CMLE, CMLT, CMTST, CNT, DUP, EOR, EXT, FCSEL, INS, MLA, MLS, MOV, MOVI, MUL, MVN, MVNI, NEG, NOT, ORN, ORR, PMUL, PMULL, PMULL2, RADDHN, RADDHN2, RBIT, REV16, REV32, RSHRN, RSHRN2, RSUBHN, RSUBHN2, SABA, SABD, SABAL, SABAL2, SABDL, SABDL2, SADALP, SADDL, SADDL2, SADDLP, SADDLV, SADDW, SADDW2, SHADD, SHL, SHLL, SHLL2, SHR, SHRN, SHRN2, SHSUB, SLI, SMAX, SMAXP, SMAXV, SMIN, SMINP, SMINV, SMLAL, SMLAL2, SMLS, SMLS2, SMOV, SMULL, SMULL2, SRI, SSHL, SSHLL, SSHLL2, SSHR, SSRA, SSUBL, SSUBL2, SSUBW, SSUBW2, SUB, SUBHN, SUBHN2, SXTL, SXTL2, TBL, TBX, TRN1, TRN2, UABA, UABAL, UABAL2, UABD, UABDL, UABDL2, UADALP, UADDL, UADDL2, UADDLP, UADDLV, UADDW, UADDW2,UHADD, UHSUB, UMAX, UMAXP, UMAXV, UMIN, UMINP, UMINV, UMLAL, UMLAL2, UMLS, UMOV, UMLS2, UMULL, UMULL2, USHL, USHLL, USHLL2, USHR, USRA, USUBL, USUBL2, USUBW, USUBW2, UXTL, UXTL2, UZP1, UZP2, XTN, XTN2, ZIP1, ZIP2, FDIV, and FSQRT. <p>The architecture makes no statement about the timing properties when the PSTATE.DIT bit is not set. However, it is likely that many of these instructions have timing that is invariant of the data in many situations.</p>	0b0
[23:0]	RESO	Reserved	RESO

Access

MRS <Xt>, DIT

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b101

MSR DIT, $\langle X_t \rangle$

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b101

MSR DIT, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b010

Accessibility

MRS <Xt>, DIT

```
if PSTATE.EL == EL0 then
    X[t, 64] = Zeros(39):PSTATE.DIT:Zeros(24);
elseif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(39):PSTATE.DIT:Zeros(24);
elseif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(39):PSTATE.DIT:Zeros(24);
```

MSR DIT, <Xt>

```
if PSTATE.EL == EL0 then
    PSTATE.DIT = X[t, 64]<24>;
elseif PSTATE.EL == EL1 then
    PSTATE.DIT = X[t, 64]<24>;
elseif PSTATE.EL == EL2 then
    PSTATE.DIT = X[t, 64]<24>;
```

MSR DIT, #<imm>

A.2.2.74 SSBS, Speculative Store Bypass Safe

Allows access to the Speculative Store Bypass Safe bit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

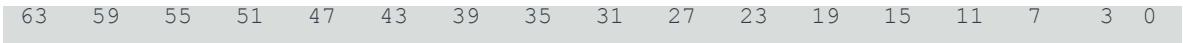
Generic System Control

Access type

See bit descriptions

Reset value

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-134: AARCH64_SSBS bit assignments

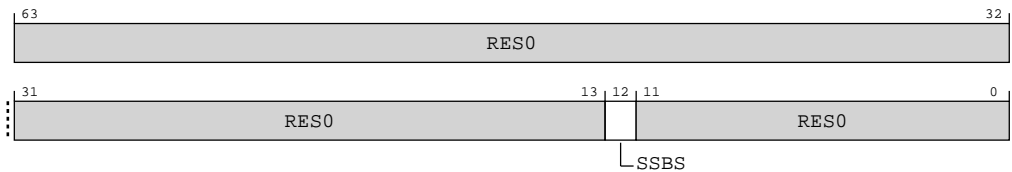


Table A-342: SSBS bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	RES0
[12]	SSBS	<p>Speculative Store Bypass Safe.</p> <p>Prohibits speculative loads or stores which might practically allow a cache timing side channel.</p> <p>A cache timing side channel might be exploited where a load or store uses an address that is derived from a register that is being loaded from memory using a load instruction speculatively read from a memory location. If PSTATE.SSBS is enabled, the address derived from the load instruction might be from earlier in the coherence order than the latest store to that memory location with the same virtual address.</p> <p>0b0</p> <p>Hardware is not permitted to load or store speculatively, in a manner that could practically give rise to a cache timing side channel, using an address derived from a register value that has been loaded from memory using a load instruction (L) that speculatively reads an entry from earlier in the coherence order from that location being loaded from than the entry generated by the latest store (S) to that location using the same virtual address as L.</p> <p>0b1</p> <p>Hardware is permitted to load or store speculatively, in a manner that could practically give rise to a cache timing side channel, using an address derived from a register value that has been loaded from memory using a load instruction (L) that speculatively reads an entry from earlier in the coherence order fro that location being loaded from than the entry generated by the latest store (S) to that location using the same virtual address as L.</p> <p>The value of this bit is set to the value in the SCTLR_ELx.DSSBS field on taking an exception to ELx.</p>	x
[11:0]	RES0	Reserved	RES0

Access

MRS <Xt>, SSBS

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b110

MSR SSBS, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b110

MSR SSBS, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b001

Accessibility

MRS <Xt>, SSBS

```
if PSTATE.EL == EL0 then
    X[t, 64] = Zeros(51):PSTATE.SSBS:Zeros(12);
elsif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(51):PSTATE.SSBS:Zeros(12);
elsif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(51):PSTATE.SSBS:Zeros(12);
```

MSR SSBS, <Xt>

```
if PSTATE.EL == EL0 then
    PSTATE.SSBS = X[t, 64]<12>;
elsif PSTATE.EL == EL1 then
    PSTATE.SSBS = X[t, 64]<12>;
elsif PSTATE.EL == EL2 then
    PSTATE.SSBS = X[t, 64]<12>;
```

MSR SSBS, #<imm>

A.2.2.75 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

RAZWI

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx00	x000	0xx0	0000	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-135: AARCH64_FPCR bit assignments

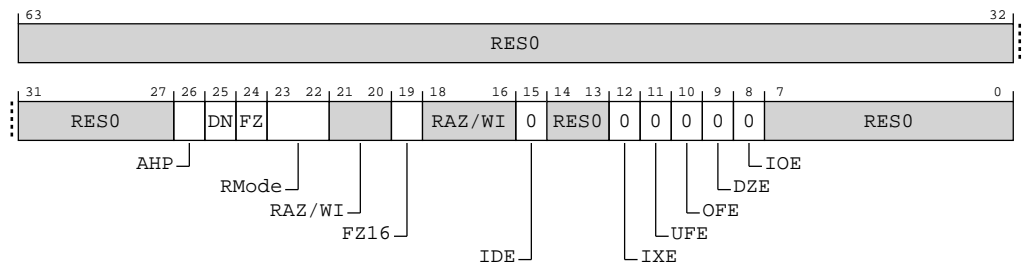


Table A-346: FPCR bit descriptions

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

Bits	Name	Description	Reset
[25]	DN	<p>Default NaN use for NaN propagation.</p> <p>0b0 NaN operands propagate through to the output of a floating-point operation.</p> <p>0b1 Any operation involving one or more NaNs returns the Default NaN.</p> <p>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.</p> <p>The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.</p>	x
[24]	FZ	<p>Flushing denormalized numbers to zero control bit.</p> <p>0b0 Flushing denormalized numbers to zero disabled.</p> <p>0b1 Flushing denormalized numbers to zero enabled.</p> <p>For more information, see <i>Flushing denormalized numbers to zero</i> in the Arm® Architecture Reference Manual for A-profile architecture and the pseudocode of the floating-point instructions.</p>	x
[23:22]	RMode	<p>Rounding Mode control field.</p> <p>0b00 Round to Nearest (RN) mode.</p> <p>0b01 Round towards Plus Infinity (RP) mode.</p> <p>0b10 Round towards Minus Infinity (RM) mode.</p> <p>0b11 Round towards Zero (RZ) mode.</p> <p>The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.</p>	xx
[21:20]	RAZ/WI	Reserved	RAZ/ WI
[19]	FZ16	<p>When IsFeatureImplemented(FEAT_FP16) Flushing denormalized numbers to zero control bit on half-precision data-processing instructions.</p> <p>0b0 For some instructions, this bit disables flushing to zero of inputs and outputs that are half-precision denormalized numbers.</p> <p>0b1 Flushing denormalized numbers to zero enabled.</p> <p>The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations.</p> <p>For more information, see <i>Flushing denormalized numbers to zero</i> in the Arm® Architecture Reference Manual for A-profile architecture and the pseudocode of the floating-point instructions.</p> <p>Otherwise RES0</p>	xxxx

Bits	Name	Description	Reset
[18:16]	RAZ/WI	Reserved	RAZ/WI
[15]	IDE	Input Denormal floating-point exception trap enable. 0b0 Untrapped exception handling selected. If the floating-point exception occurs, the AArch64-FPSR.IDC bit is set to 1.	0b0
[14:13]	RES0	Reserved	RES0
[12]	IXE	Inexact floating-point exception trap enable. 0b0 Untrapped exception handling selected. If the floating-point exception occurs, the AArch64-FPSR.IXC bit is set to 1.	0b0
[11]	UFE	Underflow floating-point exception trap enable. 0b0 Untrapped exception handling selected. If the floating-point exception occurs, the AArch64-FPSR.UFC bit is set to 1.	0b0
[10]	OFE	Overflow floating-point exception trap enable. 0b0 Untrapped exception handling selected. If the floating-point exception occurs, the AArch64-FPSR.OFC bit is set to 1.	0b0
[9]	DZE	Divide by Zero floating-point exception trap enable. 0b0 Untrapped exception handling selected. If the floating-point exception occurs, the AArch64-FPSR.DZC bit is set to 1.	0b0
[8]	IOE	Invalid Operation floating-point exception trap enable. 0b0 Untrapped exception handling selected. If the floating-point exception occurs, the AArch64-FPSR.IOC bit is set to 1.	0b0
[7:0]	RES0	Reserved	RES0

Access

MRS <Xt>, FPCR

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

MSR FPCR, <Xt>

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

Accessibility

MRS <Xt>, FPCR

```
if PSTATE.EL == EL0 then
    if CPACR_EL1.FPEN != '11' then
```

```

        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
        elsif CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            X[t, 64] = FPCR;
    elsif PSTATE.EL == EL1 then
        if CPACR_EL1.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL1, 0x07);
        elsif CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            X[t, 64] = FPCR;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            X[t, 64] = FPCR;

```

MSR FPCR, <Xt>

```

    if PSTATE.EL == EL0 then
        if CPACR_EL1.FPEN != '11' then
            if HCR_EL2.TGE == '1' then
                AArch64.SystemAccessTrap(EL2, 0x00);
            else
                AArch64.SystemAccessTrap(EL1, 0x07);
        elsif CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            FPCR = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if CPACR_EL1.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL1, 0x07);
        elsif CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            FPCR = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            FPCR = X[t, 64];

```

A.2.2.76 FPSR, Floating-point Status Register

Provides floating-point system status information.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

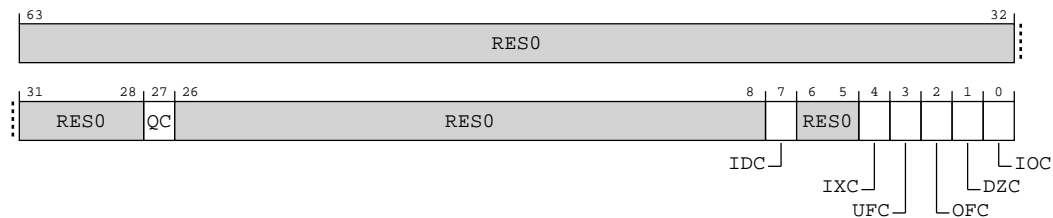
See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-136: AARCH64_FPSR bit assignments****Table A-349: FPSR bit descriptions**

Bits	Name	Description	Reset
[63:28]	RES0	Reserved	RES0
[27]	QC	Cumulative saturation bit, Advanced SIMD only. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since 0 was last written to this bit.	x
[26:8]	RES0	Reserved	RES0
[7]	IDC	Input Denormal cumulative floating-point exception bit. This bit is set to 1 to indicate that the Input Denormal floating-point exception has occurred since 0 was last written to this bit. How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.IDE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.IDE is 0.	x
[6:5]	RES0	Reserved	RES0
[4]	IXC	Inexact cumulative floating-point exception bit. This bit is set to 1 to indicate that the Inexact floating-point exception has occurred since 0 was last written to this bit. How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.IXE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.IXE is 0. The criteria for the Inexact floating-point exception to occur are affected by whether denormalized numbers are flushed to zero and by the value of the AArch64-FPCR.AH bit. For more information, see <i>Floating-point exceptions and exception traps</i> in the Arm® Architecture Reference Manual for A-profile architecture .	x

Bits	Name	Description	Reset
[3]	UFC	Underflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Underflow floating-point exception has occurred since 0 was last written to this bit. How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.UFE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.UFE is 0 or if flushing denormalized numbers to zero is enabled. The criteria for the Underflow floating-point exception to occur are affected by whether denormalized numbers are flushed to zero and by the value of the AArch64-FPCR.AH bit. For more information, see <i>Floating-point exceptions and exception traps</i> in the Arm® Architecture Reference Manual for A-profile architecture .	x
[2]	OFC	Overflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Overflow floating-point exception has occurred since 0 was last written to this bit. How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.OFE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.OFE is 0.	x
[1]	DZC	Divide by Zero cumulative floating-point exception bit. This bit is set to 1 to indicate that the Divide by Zero floating-point exception has occurred since 0 was last written to this bit. How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.DZE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.DZE is 0.	x
[0]	IOC	Invalid Operation cumulative floating-point exception bit. This bit is set to 1 to indicate that the Invalid Operation floating-point exception has occurred since 0 was last written to this bit. How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the AArch64-FPCR.IOE bit. This bit is set to 1 to indicate a floating-point exception only if AArch64-FPCR.IOE is 0. The criteria for the Invalid Operation floating-point exception to occur are affected by the value of the AArch64-FPCR.AH bit. For more information, see <i>Floating-point exceptions and exception traps</i> in the Arm® Architecture Reference Manual for A-profile architecture .	x

Access

MRS <Xt>, FPSR

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

MSR FPSR, <Xt>

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

Accessibility

MRS <Xt>, FPSR

```

if PSTATE.EL == EL0 then
    if CPACR_EL1.FPEN != '11' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
        elsif CPTR_EL2.TFP == '1' then

```

```

        AArch64.SystemAccessTrap(EL2, 0x07);
    else
        X[t, 64] = FPSR;
    elsif PSTATE.EL == EL1 then
        if CPACR_EL1.FPEN == 'x0' then
            AArch64.SystemAccessTrap(EL1, 0x07);
        elsif CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            X[t, 64] = FPSR;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TFP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x07);
        else
            X[t, 64] = FPSR;

```

MSR FPSR, <Xt>

```

if PSTATE.EL == EL0 then
    if CPACR_EL1.FPEN != '11' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    else
        FPSR = X[t, 64];
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    else
        FPSR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    else
        FPSR = X[t, 64];

```

A.2.2.77 TPIDR_ELO, ELO Read/Write Software Thread ID Register

Provides a location where software executing at EL0 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

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-137: AARCH64_TPIDR_ELO bit assignments

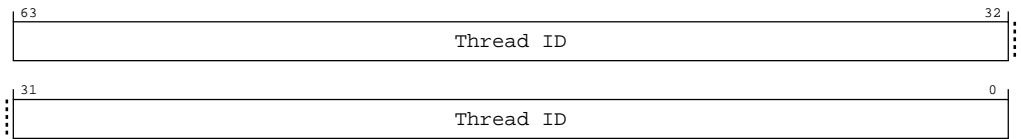


Table A-352: TPIDR_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Thread ID. Thread identifying information stored by software running at this Exception level.	64 {x}

Access

MRS <Xt>, TPIDR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b010

MSR TPIDR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1101	0b0000	0b010

Accessibility

MRS <Xt>, TPIDR_ELO

```
if PSTATE.EL == EL0 then
    X[t, 64] = TPIDR_ELO;
elseif PSTATE.EL == EL1 then
    X[t, 64] = TPIDR_ELO;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TPIDR_ELO;
```

MSR TPIDR_ELO, <Xt>

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

A.2.2.78 TPIDRRO_ELO, ELO Read-Only Software Thread ID Register

Provides a location where software executing at EL1 or higher can store thread identifying information that is visible to software executing at EL0, for OS management purposes.

The PE makes no use of this register.

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-138: AARCH64_TPIDRRO_ELO bit assignments

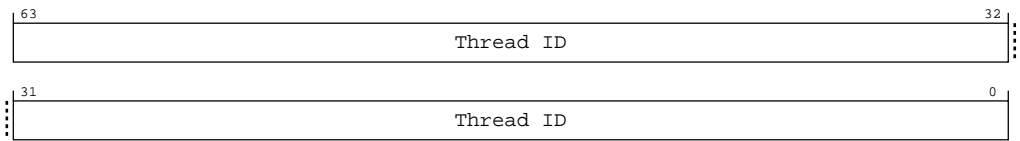


Table A-355: TPIDRRO_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Thread ID. Thread identifying information stored by software running at this Exception level.	64 {x}

Access

MRS <Xt>, TPIDRRO_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b11101	0b0000	0b011

MSR TPIDRRO_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b11101	0b0000	0b011

Accessibility

MRS <Xt>, TPIDRRO_ELO

```

if PSTATE.EL == EL0 then
    X[t, 64] = TPIDRRO_ELO;
elseif PSTATE.EL == EL1 then
    X[t, 64] = TPIDRRO_ELO;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TPIDRRO_ELO;

```

MSR TPIDRRO_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    TPIDRRO_ELO = X[t, 64];
elseif PSTATE.EL == EL2 then
    TPIDRRO_ELO = X[t, 64];

```

A.2.2.79 MPUIR_EL2, MPU Type Register (EL2)

Identifies the number of regions supported by the EL2 MPU.

Configurations

This register is available in all configurations.

Attributes**Width**

64

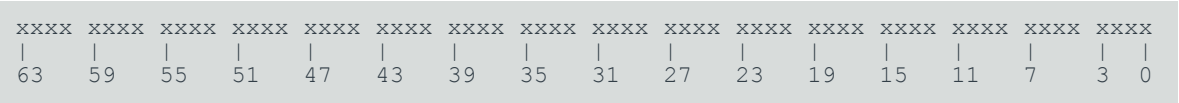
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-139: AARCH64_MPUIR_EL2 bit assignments

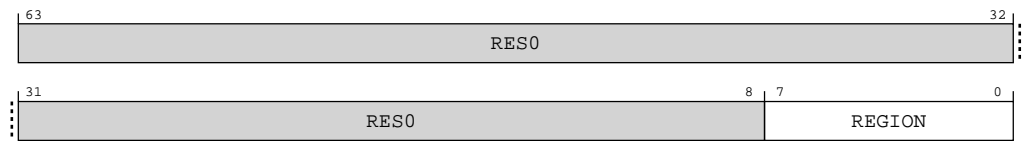


Table A-358: MPUIR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	REGION	The number of EL2 MPU regions supported.	8 {x}

Access

MRS <Xt>, MPUIR_EL2

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

Accessibility

MRS <Xt>, MPUIR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MPUIR_EL2;
```

A.2.2.80 ACTLR_EL2, Auxiliary Control Register (EL2)

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

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	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-140: AARCH64_ACTLR_EL2 bit assignments

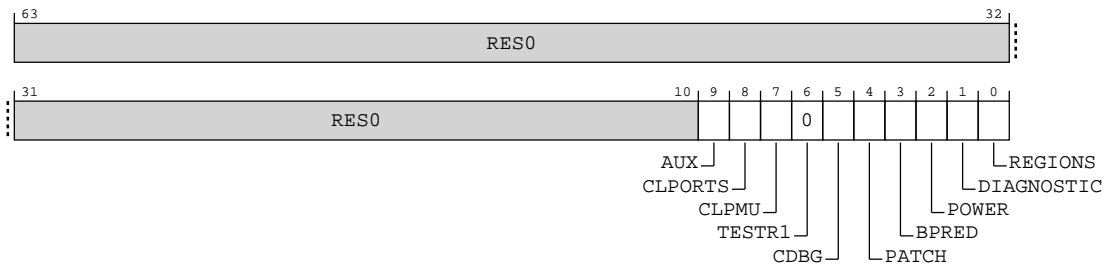


Table A-360: ACTLR_EL2 bit descriptions

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

Bits	Name	Description	Reset
[9]	AUX	<p>Auxiliary registers write control.</p> <p>0b0</p> <p>Write accesses from EL1 to AArch64-IMP_CPUACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 are trapped to EL2.</p> <p>0b1</p> <p>Write accesses from EL1 to AArch64-IMP_CPUACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 are allowed.</p>	0b0
[8]	CLPORTS	<p>Cluster L2 partitioning and ports registers write control.</p> <p>0b0</p> <p>Write accesses from EL1 to AArch64-IMP_CLUSTERQOSR_EL1, AArch64-IMP_CLUSTERACELSLTLR_EL1, AArch64-IMP_CLUSTERSID_EL1, AArch64-IMP_CLUSTERACPSID_EL1 and AArch64-IMP_CLUSTERPARTCR_EL1 are trapped to EL2.</p> <p>0b1</p> <p>Write accesses from EL1 to AArch64-IMP_CLUSTERQOSR_EL1, AArch64-IMP_CLUSTERACELSLTLR_EL1, AArch64-IMP_CLUSTERSID_EL1, AArch64-IMP_CLUSTERACPSID_EL1 and AArch64-IMP_CLUSTERPARTCR_EL1 are allowed.</p>	0b0
[7]	CLPMU	<p>Cluster Performance Monitor Unit registers write control.</p> <p>0b0</p> <p>Write accesses from EL1 to AArch64-IMP_CLUSTERPMCR_EL1, AArch64-IMP_CLUSTERPMCNTENSET_EL1, AArch64-IMP_CLUSTERPMCNTENCLR_EL1, AArch64-IMP_CLUSTERPMOVSSSET_EL1, AArch64-IMP_CLUSTERPMOVSLR_EL1, AArch64-IMP_CLUSTERPMSELR_EL1, AArch64-IMP_CLUSTERPMINTENSET_EL1, AArch64-IMP_CLUSTERPMINTENCLR_EL1, AArch64-IMP_CLUSTERPMCCNTR_EL1, AArch64-IMP_CLUSTERPMXEVTYPYPER_EL1, AArch64-IMP_CLUSTERPMXEVNTR_EL1, AArch64-IMP_CLUSTERPMCLAIMCLR_EL1 and AArch64-IMP_CLUSTERPMCLAIMSET_EL1 are trapped to EL2.</p> <p>0b1</p> <p>Write accesses from EL1 to AArch64-IMP_CLUSTERPMCR_EL1, AArch64-IMP_CLUSTERPMCNTENSET_EL1, AArch64-IMP_CLUSTERPMCNTENCLR_EL1, AArch64-IMP_CLUSTERPMOVSSSET_EL1, AArch64-IMP_CLUSTERPMOVSLR_EL1, AArch64-IMP_CLUSTERPMSELR_EL1, AArch64-IMP_CLUSTERPMINTENSET_EL1, AArch64-IMP_CLUSTERPMINTENCLR_EL1, AArch64-IMP_CLUSTERPMCCNTR_EL1, AArch64-IMP_CLUSTERPMXEVTYPYPER_EL1, AArch64-IMP_CLUSTERPMXEVNTR_EL1, AArch64-IMP_CLUSTERPMCLAIMCLR_EL1 and AArch64-IMP_CLUSTERPMCLAIMSET_EL1 are allowed.</p>	0b0
[6]	TESTR1	0b0	0b0
[5]	CDBG	<p>Cache debug registers read/write control.</p> <p>0b0</p> <p>Read accesses from EL1 to AArch64-IMP_CDBGDR0_EL1, AArch64-IMP_CDBGDR1_EL1 and AArch64-IMP_CLUSTERCDBGDR0_EL1, as well as executing from EL1 SYS IMP_CDBGDCT, SYS IMP_CDBGICT, SYS IMP_CDBGTT, SYS IMP_CDBGDCD, SYS IMP_CDBGICD, SYS IMP_CDBGTD, SYS IMP_CLUSTERCDBGL2D, SYS IMP_CLUSTERCDBGL2DT, SYS IMP_CLUSTERCDBGL2T, SYS IMP_CLUSTERCDBGLCUDT are trapped to EL2.</p> <p>0b1</p> <p>Read accesses from EL1 to AArch64-IMP_CDBGDR0_EL1, AArch64-IMP_CDBGDR1_EL1 and AArch64-IMP_CLUSTERCDBGDR0_EL1, as well as executing from EL1 SYS IMP_CDBGDCT, SYS IMP_CDBGICT, SYS IMP_CDBGTT, SYS IMP_CDBGDCD, SYS IMP_CDBGICD, SYS IMP_CDBGTD, SYS IMP_CLUSTERCDBGL2D, SYS IMP_CLUSTERCDBGL2DT, SYS IMP_CLUSTERCDBGL2T, SYS IMP_CLUSTERCDBGLCUDT are allowed.</p>	0b0

Bits	Name	Description	Reset
[4]	PATCH	<p>Patch registers read/write control.</p> <p>0b0</p> <p>Read and write accesses from EL1 to AArch64-IMP_CPUPSELR_EL1, AArch64-IMP_CPUPCR_EL1, AArch64-IMP_CPUPOR_EL1 and AArch64-IMP_CPUPMR_EL1 are trapped to EL2.</p> <p>0b1</p> <p>Read and write accesses from EL1 to AArch64-IMP_CPUPSELR_EL1, AArch64-IMP_CPUPCR_EL1, AArch64-IMP_CPUPOR_EL1 and AArch64-IMP_CPUPMR_EL1 are allowed.</p>	0b0
[3]	BPRED	<p>Branch prediction registers write control.</p> <p>0b0</p> <p>Write accesses from EL1 to AArch64-IMP_BPCTLR_EL1 as well as executing from EL1 SYS-IMP_BPI are trapped to EL2.</p> <p>0b1</p> <p>Write accesses from EL1 to AArch64-IMP_BPCTLR_EL1 as well as executing from EL1 SYS-IMP_BPI are allowed.</p>	0b0
[2]	POWER	<p>Power registers write control.</p> <p>0b0</p> <p>Write accesses from EL1 to AArch64-IMP_CPUPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRCTLR_EL1 and AArch64-IMP_CLUSTERPWRDN_EL1 are trapped to EL2.</p> <p>0b1</p> <p>Write accesses from EL1 to AArch64-IMP_CPUPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRCTLR_EL1 and AArch64-IMP_CLUSTERPWRDN_EL1 are allowed.</p>	0b0
[1]	DIAGNOSTIC	<p>Diagnostic registers write control.</p> <p>0b0</p> <p>Write accesses from EL1 to AArch64-IMP_MEMPROTCTLR_EL1 and AArch64-IMP_CLUSTERMEMPROTCTLR_EL1 are trapped to EL2.</p> <p>0b1</p> <p>Write accesses from EL1 to AArch64-IMP_MEMPROTCTLR_EL1 and AArch64-IMP_CLUSTERMEMPROTCTLR_EL1 are allowed.</p>	0b0
[0]	REGIONS	<p>Region registers write control.</p> <p>0b0</p> <p>Write accesses from EL1 to AArch64-IMP_ITCMREGIONR_EL1, AArch64-IMP_DTCMREGIONR_EL1, AArch64-IMP_LLPPREGIONR_EL1, AArch64-IMP_SPPREGIONR_EL1 and AArch64-IMP_LLRAMREGIONR_EL1 are trapped to EL2.</p> <p>0b1</p> <p>Write accesses from EL1 to AArch64-IMP_ITCMREGIONR_EL1, AArch64-IMP_DTCMREGIONR_EL1, AArch64-IMP_LLPPREGIONR_EL1, AArch64-IMP_SPPREGIONR_EL1 and AArch64-IMP_LLRAMREGIONR_EL1 are allowed.</p>	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;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ACTLR_EL2;
```

MSR ACTLR_EL2, <Xt>

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

A.2.2.81 HCR_EL2, Hypervisor Configuration Register

Provides configuration controls for virtualization, including defining whether various operations are trapped to EL2.

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	x0xx	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-141: AARCH64_HCR_EL2 bit assignments

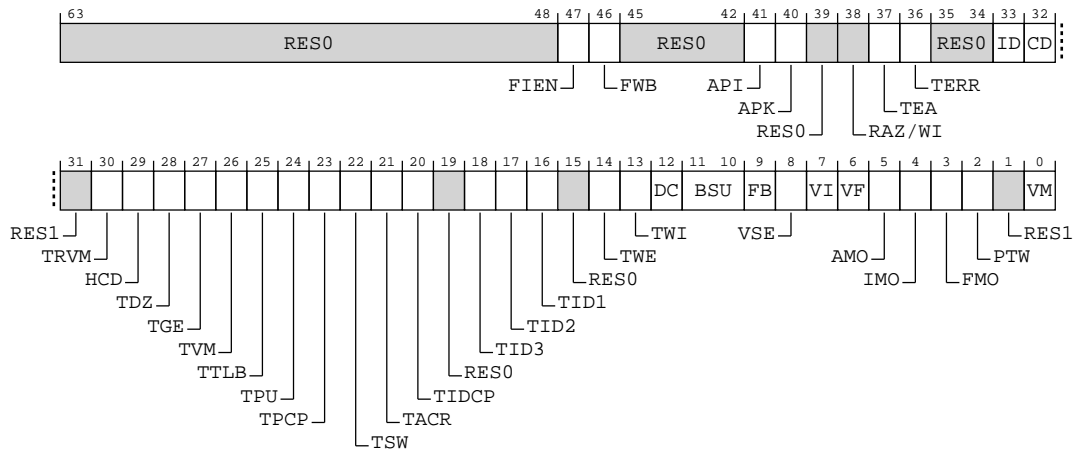


Table A-363: HCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47]	FIEN	<p>Fault Injection Enable. Unless this bit is set to 1, accesses to the AArch64-ERXPFPGCDN_EL1, AArch64-ERXPFPGCTL_EL1, and AArch64-ERXPFPGF_EL1 registers from EL1 generate a Trap exception to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18.</p> <p>0b0</p> <p>Accesses to the specified registers from EL1 are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p>	x
[46]	FWB	<p>Forced Write-Back. Defines the combined cacheability attributes in a 2 stage translation regime.</p> <p>0b0</p> <p>When this bit is 0, then:</p> <ul style="list-style-type: none"> The combination of stage 1 and stage 2 translations on memory type and cacheability attributes are as described in the Armv8.0 architecture. For more information, see <i>Combining stage 1 and stage 2 memory type attributes</i> in the Arm® Architecture Reference Manual for A-profile architecture. The encoding of the stage 2 memory type and cacheability attributes is derived from AArch64-MAIR_EL2, as described in the Armv8-R AArch64 architecture. <p>0b1</p> <p>When this bit is 1, then:</p> <ul style="list-style-type: none"> The inner and outer memory attributes for stage 2 EL1&O translation regime must be the same with the same encoding, otherwise, the combined attribute is UNKNOWN. If stage 2 EL1&O translation regime memory attribute is Write-Back with AArch64-MAIR_EL2.Attr[7:6] = 0b11, then the combined attribute is Normal Write-Back. For all other encodings, the combination of stage 1 and stage 2 translations on memory type and cacheability attributes are as described in the Armv8.0 architecture. <p>This bit is permitted to be cached in a TLB.</p>	x
[45:42]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[41]	API	<p>Controls the use of instructions related to Pointer Authentication:</p> <ul style="list-style-type: none"> In EL0, the associated AArch64-SCTLR_EL1.En<N><M>==1. In EL1, the associated AArch64-SCTLR_EL1.En<N><M>==1. <p>Traps are reported using EC syndrome value 0x09. The Pointer Authentication instructions trapped are:</p> <ul style="list-style-type: none"> AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB. PACGA, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZB. RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ. ERETAA, ERETAB, LDRAA, and LDRAB. <p>0b0</p> <p>The instructions related to Pointer Authentication are trapped to EL2, when EL2 is enabled in the current Security state and the instructions are enabled for the EL1&O translation regime, from:</p> <ul style="list-style-type: none"> EL0. EL1. <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p>	x
[40]	APK	<p>Trap registers holding "key" values for Pointer Authentication. Traps accesses to the following registers from EL1 to EL2, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x18:</p> <ul style="list-style-type: none"> AArch64-APIAKeyLo_EL1, AArch64-APIAKeyHi_EL1, AArch64-APIBKeyLo_EL1, AArch64-APIBKeyHi_EL1, AArch64-APDAKeyLo_EL1, AArch64-APDAKeyHi_EL1, AArch64-APDBKeyLo_EL1, AArch64-APDBKeyHi_EL1, AArch64-APGAKeyLo_EL1, and AArch64-APGAKeyHi_EL1. <p>0b0</p> <p>Access to the registers holding "key" values for pointer authentication from EL1 are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>0b1</p> <p>This control does not cause any instructions to be trapped.</p>	x
[39]	RES0	Reserved	RES0
[38]	RAZ/WI	Reserved	RAZ/WI
[37]	TEA	<p>Route synchronous External abort exceptions to EL2.</p> <p>0b0</p> <p>This control does not cause exceptions to be routed from EL0 and EL1 to EL2.</p> <p>0b1</p> <p>Route synchronous External abort exceptions from EL0 and EL1 to EL2, when EL2 is enabled in the current Security state.</p>	x

Bits	Name	Description	Reset
[36]	TERR	<p>Trap accesses of Error Record registers. Enables a trap to EL2 on accesses of Error Record registers.</p> <p>0b0</p> <p>Accesses of the specified Error Record registers are not trapped by this mechanism.</p> <p>0b1</p> <p>Accesses of the specified Error Record registers at EL1 are trapped to EL2, unless the instruction generates a higher priority exception.</p> <p>In AArch64 state, the instructions affected by this control are:</p> <ul style="list-style-type: none"> MRS and MSR accesses to AArch64-ERRSELR_EL1, AArch64-ERXADDR_EL1, AArch64-ERXCTLR_EL1, AArch64-ERXMISCO_EL1, AArch64-ERXMISC1_EL1, and AArch64-ERXSTATUS_EL1. MRS accesses to AArch64-ERRIDR_EL1 and AArch64-ERXFR_EL1. If FEAT_RASv1p1 is implemented, MRS and MSR accesses to AArch64-ERXMISC2_EL1 and AArch64-ERXMISC3_EL1. <p>Unless the instruction generates a higher priority exception, trapped instructions generate an exception to EL2.</p> <p>Trapped AArch64 instructions are reported using EC syndrome value 0x18.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field is permitted to be RES0 if all of the following are true: <ul style="list-style-type: none"> AArch64-ERRSELR_EL1 and all ERX* registers are implemented as UNDEFINED or RAZ/WI. AArch64-ERRIDR_EL1.NUM is zero. 	x
[35:34]	RES0	Reserved	RES0
[33]	ID	<p>Stage 2 Instruction access cacheability disable. For the EL1&0 translation regime, when EL2 is enabled in the current Security state and HCR_EL2.VM==1, this control forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.</p> <p>Note:</p> <p>The behavior is the same irrespective of whether the instruction accesses are to an MPU region or Background region.</p> <p>0b0</p> <p>This control has no effect on stage 2 of the EL1&0 translation regime.</p> <p>0b1</p> <p>Forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.</p> <p>This bit has no effect on the EL2 translation regime.</p>	x

Bits	Name	Description	Reset
[32]	CD	<p>Stage 2 Data access cacheability disable. For the EL1&0 translation regime, when EL2 is enabled in the current Security state and HCR_EL2.VM==1, this control forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.</p> <p>Note: The behavior is same irrespective of whether the data accesses is to MPU region or Background region.</p> <p>0b0 This control has no effect on stage 2 of the EL1&0 translation regime for data accesses and translation table walks.</p> <p>0b1 Forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.</p> <p>This bit has no effect on the EL2 translation regime.</p>	x
[31]	RES1	Reserved	RES1
[30]	TRVM	<p>Trap Reads of Virtual Memory controls. Traps reads of the virtual memory control registers to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> If EL1 is using AArch64 state, EL1 accesses to the following registers are trapped to EL2 and reported using EC syndrome value 0x18 for MRS: <ul style="list-style-type: none"> AArch64-SCTLR_EL1, AArch64-TTBR0_EL1, AArch64-TTBR1_EL1, AArch64-TCR_EL1, AArch64-ESR_EL1, AArch64-FAR_EL1, AArch64-AFSR0_EL1, AArch64-AFSR1_EL1, AArch64-MAIR_EL1, AArch64-AMAIR_EL1, AArch64-CONTEXTIDR_EL1. If EL1 is in PMSAv8-64 context, EL1 accesses to the following registers are also trapped to EL2 and reported using EC syndrome value 0x18 - AArch64-PRENR_EL1, AArch64-PRSELR_EL1, AArch64-PRBAR_EL1, AArch64-PRBARn_EL1, AArch64-PRLAR_EL1, AArch64-PRLARn_EL1. <p>0b0 This control does not cause any instructions to be trapped.</p> <p>0b1 Read accesses to the specified Virtual Memory control registers are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p> <p>Note: EL2 provides a second stage of address translation, that a hypervisor can use to remap the address map defined by a Guest OS. In addition, a hypervisor can trap attempts by a Guest OS to write to the registers that control the memory system. A hypervisor might use this trap as part of its virtualization of memory management.</p>	x

Bits	Name	Description	Reset
[29]	HCD	<p>HVC instruction disable. Disables EL1 execution of HVC instructions, from both Execution states, when EL2 is enabled in the current Security state, reported using EC syndrome value 0x00.</p> <p>0b0</p> <p>HVC instruction execution is enabled at EL2 and EL1.</p> <p>0b1</p> <p>HVC instructions are UNDEFINED at EL2 and EL1. Any resulting exception is taken to the Exception level at which the HVC instruction is executed.</p> <p>Note:</p> <p>HVC instructions are always UNDEFINED at EL0.</p>	x
[28]	TDZ	<p>Trap DC ZVA instructions. Traps EL0 and EL1 execution of DC ZVA instructions to EL2, when EL2 is enabled in the current Security state, from AArch64 state only, reported using EC syndrome value 0x18.</p> <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>In AArch64 state, any attempt to execute an instruction this trap applies to at EL1, or at EL0 when the instruction is not UNDEFINED at EL0, is trapped to EL2 when EL2 is enabled in the current Security state.</p> <p>Reading the AArch64-DCZID_EL0 returns a value that indicates that the instructions this trap applies to are not supported.</p>	x
[27]	TGE	<p>Trap General Exceptions, from EL0.</p> <p>0b0</p> <p>This control has no effect on execution at EL0.</p> <p>0b1</p> <p>When EL2 is not enabled in the current Security state, this control has no effect on execution at EL0.</p> <p>When EL2 is enabled in the current Security state, in all cases:</p> <ul style="list-style-type: none"> • All exceptions that would be routed to EL1 are routed to EL2. • If EL1 is using AArch64, the AArch64-SCTLR_EL1.M field is treated as being 0 for all purposes other than returning the result of a direct read of AArch64-SCTLR_EL1. • If stage 1 EL1&O translation regime is in PMSAv8-64 context, the AArch64-SCTLR_EL1.BR field is treated as being 0 for all purposes other than returning the result of a direct read of AArch64-SCTLR_EL1. • All virtual interrupts are disabled. • Any IMPLEMENTATION DEFINED mechanisms for signaling virtual interrupts are disabled. • An exception return to EL1 is treated as an illegal exception return. • The AArch64-MDCR_EL2.{TDRA, TDOSA, TDA, TDE} fields are treated as being 1 for all purposes other than returning the result of a direct read of AArch64-MDCR_EL2. <p>In addition, when EL2 is enabled in the current Security state, the Effective values of the HCR_EL2.{FMO, IMO, AMO} fields are 1.</p> <p>HCR_EL2.TGE must not be cached in a TLB.</p>	x

Bits	Name	Description	Reset
[26]	TVM	<p>Trap Virtual Memory controls. Traps writes to the virtual memory control registers to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> If EL1 is using AArch64 state, the following registers are trapped to EL2 and reported using EC syndrome value 0x18 for MSR: <ul style="list-style-type: none"> AArch64-SCTLR_EL1, AArch64-TTBRO_EL1, AArch64-TTBR1_EL1, AArch64-TCR_EL1, AArch64-ESR_EL1, AArch64-FAR_EL1, AArch64-AFSR0_EL1, AArch64-AFSR1_EL1, AArch64-MAIR_EL1, AArch64-AMAIR_EL1, AArch64-CONTEXTIDR_EL1. If EL1 is in PMSAv8-64 context, EL1 accesses to the following registers are also trapped to EL2 and reported using EC syndrome value 0x18 - AArch64-PRENR_EL1, AArch64-PRSELR_EL1, AArch64-PRBAR_EL1, AArch64-PRBARn_EL1, AArch64-PRLAR_EL1, AArch64-PRLARn_EL1. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>Write accesses to the specified Virtual Memory control registers are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p>	x
[25]	TTLB	<p>Trap TLB maintenance instructions. Traps EL1 execution of TLB maintenance instructions to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> When EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18: <ul style="list-style-type: none"> TLBI VMALLE1, TLBI VAE1, TLBI ASIDE1, TLBI VAAE1, TLBI VALE1, TLBI VAALE1. TLBI VMALLE1IS, TLBI VAE1IS, TLBI ASIDE1IS, TLBI VAAE1IS, TLBI VALE1IS, TLBI VAALE1IS. If FEAT_TLBIOS is implemented, this trap applies to TLBI VMALLE1OS, TLBI VAE1OS, TLBI ASIDE1OS, TLBI VAAE1OS, TLBI VALE1OS, TLBI VAALE1OS. If FEAT_TLBIRANGE is implemented, this trap applies to TLBI RVAE1, TLBI RVAAE1, TLBI RVALE1, TLBI RVALE1IS, TLBI RVAE1IS, TLBI RVAAE1IS, TLBI RVALE1IS, TLBI RVALE1IS. If FEAT_TLBIOS and FEAT_TLBIRANGE are implemented, this trap applies to TLBI RVAE1OS, TLBI RVAE1OS, TLBI RVALE1OS, TLBI RVALE1OS. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>EL1 execution of the specified TLB maintenance instructions are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p> <p>Note: The TLB maintenance instructions are UNDEFINED at EL0.</p>	x

Bits	Name	Description	Reset
[24]	TPU	<p>Trap cache maintenance instructions that operate to the Point of Unification. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:</p> <ul style="list-style-type: none"> If EL0 is using AArch64 state and the value of AArch64-SCTLR_EL1.UCI is not 0, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18: <ul style="list-style-type: none"> IC IVAU, DC CVAU. If the value of AArch64-SCTLR_EL1.UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2. If EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported with EC syndrome value 0x18: <ul style="list-style-type: none"> IC IVAU, IC IALLU, IC IALLUIS, DC CVAU. <p>Note: An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:</p> <ul style="list-style-type: none"> IC IALLUIS and IC IALLU are always UNDEFINED at EL0 using AArch64. <p>0b0 This control does not cause any instructions to be trapped.</p> <p>0b1 Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.</p>	x
[23]	TPCP	<p>Trap data or unified cache maintenance instructions that operate to the Point of Coherency or Persistence. Traps execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state as follows:</p> <ul style="list-style-type: none"> If EL0 is using AArch64 state and the value of AArch64-SCTLR_EL1.UCI is not 0, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18: <ul style="list-style-type: none"> DC CIVAC, DC CVAC, DC CVAP. If the value of AArch64-SCTLR_EL1.UCI is 0 these instructions are UNDEFINED at EL0 and any resulting exception is higher priority than this trap to EL2. If EL1 is using AArch64 state, the following instructions are trapped to EL2 and reported using EC syndrome value 0x18: <ul style="list-style-type: none"> DC IVAC, DC CIVAC, DC CVAC, DC CVAP. <p>If FEAT_DPB2 is implemented, this trap also applies to DC CVADP.</p> <p>Note:</p> <ul style="list-style-type: none"> An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition: <ul style="list-style-type: none"> AArch64 instructions which invalidate by VA to the Point of Coherency are always UNDEFINED at EL0 using AArch64. In Armv8.0 and Armv8.1, this field is named TPC. From Armv8.2, it is named TPCP. <p>0b0 This control does not cause any instructions to be trapped.</p> <p>0b1 Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.</p>	x

Bits	Name	Description	Reset
[22]	TSW	<p>Trap data or unified cache maintenance instructions that operate by Set/Way. Traps execution of those cache maintenance instructions at EL1 to EL2, when EL2 is enabled in the current Security state as follows:</p> <ul style="list-style-type: none"> If EL1 is using AArch64 state, accesses to DC ISW, DC CSW, DC CISW are trapped to EL2, reported using EC syndrome value 0x18. <p>Note: An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.</p> <p>0b0 This control does not cause any instructions to be trapped.</p> <p>0b1 Execution of the specified instructions is trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p>	x
[21]	TACR	<p>Trap Auxiliary Control Registers. Traps EL1 accesses to the Auxiliary Control Registers to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> If EL1 is using AArch64 state, accesses to AArch64-ACTLR_EL1 to EL2, are trapped to EL2 and reported using EC syndrome value 0x18. <p>0b0 This control does not cause any instructions to be trapped.</p> <p>0b1 EL1 accesses to the specified registers are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p> <p>AArch64-ACTLR_EL1 is not accessible at EL0.</p> <p>The Auxiliary Control Registers are IMPLEMENTATION DEFINED registers that might implement global control bits for the PE.</p>	x

Bits	Name	Description	Reset
[20]	TIDCP	<p>Trap IMPLEMENTATION DEFINED functionality. Traps EL1 accesses to the encodings reserved for IMPLEMENTATION DEFINED functionality to EL2, when EL2 is enabled in the current Security state as follows:</p> <ul style="list-style-type: none"> In AArch64 state, access to any of the encodings in the following reserved encoding spaces are trapped and reported using EC syndrome 0x18: <ul style="list-style-type: none"> IMPLEMENTATION DEFINED System instructions, which are accessed using SYS and SYSL, with CRn == {11, 15}. IMPLEMENTATION DEFINED System registers, which are accessed using MRS and MSR with the S3_<op1>_<Cn>_<Cm>_<op2> register name. <p>When this functionality is accessed from EL0:</p> <ul style="list-style-type: none"> If HCR_EL2.TIDCP is 1, it is IMPLEMENTATION DEFINED whether any accesses from EL0 are trapped to EL2. In this implementation, such accesses are not trapped to EL2. If HCR_EL2.TIDCP is 0, any accesses from EL0 are UNDEFINED and generate an exception that is taken to EL1 or EL2. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>EL1 accesses to or execution of the specified encodings reserved for IMPLEMENTATION DEFINED functionality are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>An implementation can also include IMPLEMENTATION DEFINED registers that provide additional controls, to give finer-grained control of the trapping of IMPLEMENTATION DEFINED features.</p> <p>The trapping of accesses to these registers from EL1 is higher priority than an exception resulting from the register access being UNDEFINED.</p>	x
[19]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[18]	TID3	<p>Trap ID group 3. Traps EL1 reads of group 3 ID registers to EL2, when EL2 is enabled in the current Security state, as follows:</p> <p>In AArch64 state:</p> <ul style="list-style-type: none"> Reads of the following registers are trapped to EL2, reported using EC syndrome value 0x18: <ul style="list-style-type: none"> AArch64-ID_PFR0_EL1, AArch64-ID_PFR1_EL1, AArch64-ID_PFR2_EL1, AArch64-ID_DFR0_EL1, AArch64-ID_AFR0_EL1, AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR2_EL1, AArch64-ID_MMFR3_EL1, AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, AArch64-ID_ISAR5_EL1, AArch64-MVFR0_EL1, AArch64-MVFR1_EL1, AArch64-MVFR2_EL1. AArch64-ID_AA64PFR0_EL1, AArch64-ID_AA64PFR1_EL1, AArch64-ID_AA64DFR0_EL1, AArch64-ID_AA64DFR1_EL1, AArch64-ID_AA64ISAR0_EL1, AArch64-ID_AA64ISAR1_EL1, AArch64-ID_AA64MMFR0_EL1, AArch64-ID_AA64MMFR1_EL1, AArch64-ID_AA64AFR0_EL1, AArch64-ID_AA64AFR1_EL1. AArch64-ID_MMFR4_EL1. AArch64-ID_AA64MMFR2_EL1, AArch64-ID_ISAR6_EL1. Otherwise, it is IMPLEMENTATION DEFINED whether this field traps MRS accesses to registers in the following range that are not already mentioned in this field description: Op0 == 3, op1 == 0, CRn == c0, CRm == {c2-c7}, op2 == {0-7}. In this implementation, this field does not trap such accesses. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>The specified EL1 read accesses to ID group 3 registers are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p>	x
[17]	TID2	<p>Trap ID group 2. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> If EL1 is using AArch64, reads of AArch64-CTR_ELO, AArch64-CCSIDR_EL1, AArch64-CCSIDR2_EL1, AArch64-CLIDR_EL1, and AArch64-CSSELR_EL1 are trapped to EL2, reported using EC syndrome value 0x18. If ELO is using AArch64 and the value of AArch64-SCTLR_EL1.UCT is not 0, reads of AArch64-CTR_ELO are trapped to EL2, reported using EC syndrome value 0x18. If the value of AArch64-SCTLR_EL1.UCT is 0, then ELO reads of AArch64-CTR_ELO are trapped to EL1 and the resulting exception takes precedence over this trap. If EL1 is using AArch64, writes to AArch64-CSSELR_EL1 are trapped to EL2, reported using EC syndrome value 0x18. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>The specified EL1 and ELO accesses to ID group 2 registers are trapped to EL2, when EL2 is enabled in the current Security state.</p>	x

Bits	Name	Description	Reset
[16]	TID1	<p>Trap ID group 1. Traps EL1 reads of the following registers to EL2, when EL2 is enabled in the current Security state as follows:</p> <ul style="list-style-type: none"> Accesses of AArch64-MPUIR_EL1, AArch64-REVIDR_EL1, AArch64-AIDR_EL1, reported using EC syndrome value 0x18. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>The specified EL1 read accesses to ID group 1 registers are trapped to EL2, when EL2 is enabled in the current Security state.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p>	x
[15]	RES0	Reserved	RES0
[14]	TWE	<p>Traps EL0 and EL1 execution of WFE instructions to EL2, when EL2 is enabled in the current Security state, from both Execution states, reported using EC syndrome value 0x01.</p> <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>Any attempt to execute a WFE instruction at EL0 or EL1 is trapped to EL2, when EL2 is enabled in the current Security state, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by AArch32-SCTLR.nTWE or AArch64-SCTLR_EL1.nTWE.</p> <p>Note:</p> <p>Since a WFE can complete at any time, even without a Wakeup event, the traps on WFE are not guaranteed to be taken, even if the WFE is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.</p> <p>For more information about when WFE instructions can cause the PE to enter a low-power state, see <i>Wait for Event mechanism and Send event</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x
[13]	TWI	<p>Traps EL0 and EL1 execution of WFI instructions to EL2, when EL2 is enabled in the current Security state, from both Execution states, reported using EC syndrome value 0x01.</p> <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>Any attempt to execute a WFI instruction at EL0 or EL1 is trapped to EL2, when EL2 is enabled in the current Security state, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by AArch32-SCTLR.nTWI or AArch64-SCTLR_EL1.nTWI.</p> <p>Note:</p> <p>Since a WFI can complete at any time, even without a Wakeup event, the traps on WFI are not guaranteed to be taken, even if the WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.</p> <p>For more information about when WFI instructions can cause the PE to enter a low-power state, see <i>Wait for Interrupt</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x

Bits	Name	Description	Reset
[12]	DC	<p>Default Cacheability.</p> <p>0b0</p> <p>This control has no effect on the EL1&O translation regime.</p> <p>0b1</p> <p>In both Security states:</p> <ul style="list-style-type: none"> When EL1 is using AArch64, the PE behaves as if the value of the AArch64-SCTLR_EL1.M field is 0 for all purposes other than returning the value of a direct read of AArch64-SCTLR_EL1. If stage 1 EL1&O translation regime is in PMSAv8-64 context, the PE behaves as if the value of the AArch64-SCTLR_EL1.BR field is 0 for all purposes other than returning the value of a direct read of AArch64-SCTLR_EL1. The PE behaves as if the value of the HCR_EL2.VM field is 1 for all purposes other than returning the value of a direct read of HCR_EL2. The memory type produced by stage 1 of the EL1&O translation regime is Normal Non-Shareable, Inner Write-Back Read-Allocate Write-Allocate, Outer Write-Back Read-Allocate Write-Allocate. <p>This field has no effect on the EL2 translation regime.</p> <p>This bit is permitted to be cached in a TLB.</p>	x
[11:10]	BSU	<p>Barrier Shareability upgrade. This field determines the minimum shareability domain that is applied to any barrier instruction executed from EL1 or EL0:</p> <p>0b00</p> <p>No effect.</p> <p>0b01</p> <p>Inner Shareable.</p> <p>0b10</p> <p>Outer Shareable.</p> <p>0b11</p> <p>Full system.</p> <p>This value is combined with the specified level of the barrier held in its instruction, using the same principles as combining the shareability attributes from two stages of address translation.</p>	xx
[9]	FB	<p>Force broadcast. Causes the following instructions to be broadcast within the Inner Shareable domain when executed from EL1:</p> <p>AArch64: TLBI VMALLE1, TLBI VAE1, TLBI ASIDE1, TLBI VAAE1, TLBI VALE1, TLBI VAALE1, IC IALLU, TLBI RVAE1, TLBI RVAAE1, TLBI RVALE1, TLBI RVAALE1.</p> <p>0b0</p> <p>This field has no effect on the operation of the specified instructions.</p> <p>0b1</p> <p>When one of the specified instruction is executed at EL1, the instruction is broadcast within the Inner Shareable shareability domain.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p>	x

Bits	Name	Description	Reset
[8]	VSE	<p>Virtual SError interrupt.</p> <p>0b0</p> <p>This mechanism is not making a virtual SError interrupt pending.</p> <p>0b1</p> <p>A virtual SError interrupt is pending because of this mechanism.</p> <p>The virtual SError interrupt is enabled only when the value of HCR_EL2.{TGE, AMO} is {0, 1}.</p>	x
[7]	VI	<p>Virtual IRQ Interrupt.</p> <p>0b0</p> <p>This mechanism is not making a virtual IRQ pending.</p> <p>0b1</p> <p>A virtual IRQ is pending because of this mechanism.</p> <p>The virtual IRQ is enabled only when the value of HCR_EL2.{TGE, IMO} is {0, 1}.</p>	x
[6]	VF	<p>Virtual FIQ Interrupt.</p> <p>0b0</p> <p>This mechanism is not making a virtual FIQ pending.</p> <p>0b1</p> <p>A virtual FIQ is pending because of this mechanism.</p> <p>The virtual FIQ is enabled only when the value of HCR_EL2.{TGE, FMO} is {0, 1}.</p>	x
[5]	AMO	<p>Physical SError interrupt routing.</p> <p>0b0</p> <p>When executing at Exception levels below EL2, and EL2 is enabled in the current Security state:</p> <ul style="list-style-type: none"> When the value of HCR_EL2.TGE is 0, Physical SError interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 1, Physical SError interrupts are taken to EL2. Virtual SError interrupts are disabled. <p>0b1</p> <p>When executing at any Exception level, and EL2 is enabled in the current Security state:</p> <ul style="list-style-type: none"> Physical SError interrupts are taken to EL2. When the value of HCR_EL2.TGE is 0, then virtual SError interrupts are enabled. <p>If EL2 is enabled in the current Security state and the value of HCR_EL2.TGE is 1:</p> <ul style="list-style-type: none"> Regardless of the value of the AMO bit physical asynchronous External aborts and SError interrupts target EL2. This field behaves as 1 for all purposes other than a direct read of the value of this bit. <p>For more information, see <i>Asynchronous exception routing</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x

Bits	Name	Description	Reset
[4]	IMO	<p>Physical IRQ Routing.</p> <p>0b0</p> <p>When executing at Exception levels below EL2, and EL2 is enabled in the current Security state:</p> <ul style="list-style-type: none"> When the value of HCR_EL2.TGE is 0, Physical IRQ interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 1, Physical IRQ interrupts are taken to EL2 unless they are routed to EL3. Virtual IRQ interrupts are disabled. <p>0b1</p> <p>When executing at any Exception level, and EL2 is enabled in the current Security state:</p> <ul style="list-style-type: none"> Physical IRQ interrupts are taken to EL2, unless they are routed to EL3. When the value of HCR_EL2.TGE is 0, then Virtual IRQ interrupts are enabled. <p>If EL2 is enabled in the current Security state, and the value of HCR_EL2.TGE is 1:</p> <ul style="list-style-type: none"> Regardless of the value of the IMO bit, physical IRQ Interrupts target EL2. This field behaves as 1 for all purposes other than a direct read of the value of this bit. <p>For more information, see <i>Asynchronous exception routing</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x
[3]	FMO	<p>Physical FIQ Routing.</p> <p>0b0</p> <p>When executing at Exception levels below EL2, and EL2 is enabled in the current Security state:</p> <ul style="list-style-type: none"> When the value of HCR_EL2.TGE is 0, Physical FIQ interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 1, Physical FIQ interrupts are taken to EL2. Virtual FIQ interrupts are disabled. <p>0b1</p> <p>When executing at any Exception level, and EL2 is enabled in the current Security state:</p> <ul style="list-style-type: none"> Physical FIQ interrupts are taken to EL2. When HCR_EL2.TGE is 0, then Virtual FIQ interrupts are enabled. <p>If EL2 is enabled in the current Security state and the value of HCR_EL2.TGE is 1:</p> <ul style="list-style-type: none"> Regardless of the value of the FMO bit, physical FIQ Interrupts target EL2. This field behaves as 1 for all purposes other than a direct read of the value of this bit. <p>For more information, see <i>Asynchronous exception routing</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x

Bits	Name	Description	Reset
[2]	PTW	<p>Protected Table Walk. In the EL1&O translation regime, a translation table access made as part of a stage 1 translation table walk is subject to a stage 2 translation. The combining of the memory type attributes from the two stages of translation means the access might be made to a type of Device memory. If this occurs, then the value of this bit determines the behavior:</p> <p>0b0</p> <p>The translation table walk occurs as if it is to Normal Non-cacheable memory. This means it can be made speculatively.</p> <p>0b1</p> <p>The memory access generates a stage 2 Permission fault.</p> <p>This bit is permitted to be cached in a TLB.</p> <p>When HCR_EL2.TGE is 1, the PE ignores the value of this field for all purposes other than a direct read of this field.</p> <p>In a PMSA-only implementation, this bit is permitted to be RES0.</p>	x
[1]	RES1	Reserved	RES1
[0]	VM	<p>Virtualization enable. Enables stage 2 address translation for the EL1&O translation regime, when EL2 is enabled in the current Security state.</p> <p>0b0</p> <p>EL1&O stage 2 address translation disabled.</p> <p>0b1</p> <p>EL1&O stage 2 address translation enabled.</p> <p>If HCR_EL2.VM is 1 and SCTLR_EL2.{M, BR} is {0, 0}, then the memory attribute becomes UNKNOWN.</p> <p>When the value of this bit is 1, data cache invalidate instructions executed at EL1 perform a data cache clean and invalidate. For the invalidate by set/way instruction this behavior applies regardless of the value of the HCR_EL2.SWIO bit.</p> <p>This bit is permitted to be cached in a TLB.</p>	x

Access

MRS <Xt>, HCR_EL2

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

MSR HCR_EL2, <Xt>

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

Accessibility

MRS <Xt>, HCR_EL2

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

```
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = HCR_EL2;
```

MSR HCR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    HCR_EL2 = X[t, 64];
```

A.2.2.82 CPTR_EL2, Architectural Feature Trap Register (EL2)

Controls trapping to EL2 of accesses to AArch64-CPACR_EL1, trace, and Advanced SIMD and floating-point functionality.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

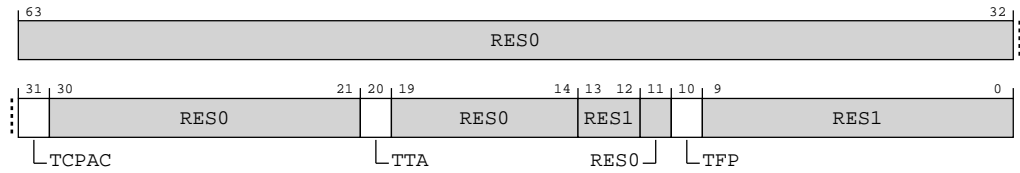
xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

This format applies in all Armv8.0 implementations.

Figure A-142: AARCH64_CPTR_EL2 bit assignments**Table A-366: CPTR_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	TCPAC	<p>In AArch64 state, traps accesses to AArch64-CPACR_EL1 from EL1 to EL2, when EL2 is enabled in the current Security state. The exception is reported using ESR_ELx.EC value 0x18.</p> <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>EL1 accesses to the following registers are trapped to EL2, when EL2 is enabled in the current Security state:</p> <ul style="list-style-type: none"> AArch64-CPACR_EL1. <p>When AArch64-HCR_EL2.TGE is 1, this control does not cause any instructions to be trapped.</p> <p>Note: AArch64-CPACR_EL1 is not accessible at ELO.</p>	x
[30:21]	RES0	Reserved	RES0
[20]	TTA	<p>Traps System register accesses to all implemented trace registers from both Execution states to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to trace registers with op0=2, op1=1, and CRn<0b1000 are trapped to EL2, reported using EC syndrome value 0x18. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>Any attempt at ELO, EL1, or EL2, to execute a System register access to an implemented trace register is trapped to EL2, when EL2 is enabled in the current Security state, unless it is trapped by one of the following controls:</p> <ul style="list-style-type: none"> AArch64-CPACR_EL1.TTA. The ETMv4 architecture does not permit ELO to access the trace registers. If the trace unit implements FEAT_ETMv4, ELO accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of AArch64-CPTR_EL2.TTA is 1. EL2 does not provide traps on trace register accesses through the optional memory-mapped interface. <p>System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.</p> <p>If System register access to the trace functionality is not supported, this bit is RES0.</p>	x

Bits	Name	Description	Reset
[19:14]	RES0	Reserved	RES0
[13:12]	RES1	Reserved	RES1
[11]	RES0	Reserved	RES0
[10]	TFP	<p>Traps execution of instructions which access the Advanced SIMD and floating-point functionality, from both Execution states to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to the following registers are trapped to EL2, reported using ESR_ELx.EC value 0x07: <ul style="list-style-type: none"> AArch64-FPCR, AArch64-FPSR, AArch64-FPEXC32_EL2, any of the SIMD and floating-point registers V0-V31, including their views as D0-D31 registers or S0-S31 registers. <p>0b0 This control does not cause execution of any instructions to be trapped.</p> <p>0b1 This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.</p> <p>AArch64-FPEXC32_EL2 is not accessible from EL0 using AArch64.</p>	x
[9:0]	RES1	Reserved	RES1

Access

MRS <Xt>, CPTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b010

MSR CPTR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b010

Accessibility

MRS <Xt>, CPTR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CPTR_EL2;

```

MSR CPTR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CPTR_EL2 = X[t, 64];

```


A.2.2.83 HACR_EL2, Hypervisor Auxiliary Control Register

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

Configurations

This register is available in all configurations.

Attributes

Width

64

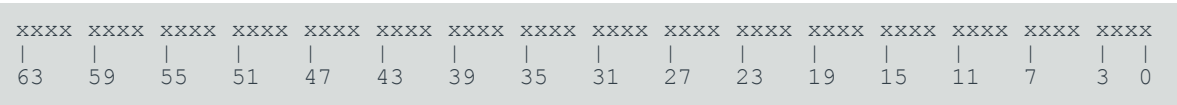
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-143: AARCH64_HACR_EL2 bit assignments

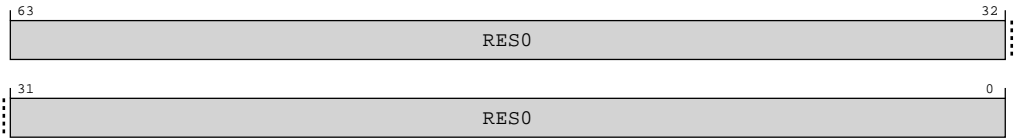


Table A-369: HACR_EL2 bit descriptions

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;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = HACR_EL2;
```

MSR HACR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    HACR_EL2 = X[t, 64];
```

A.2.2.84 VSCTLR_EL2, Virtualization System Control Register (EL2)

Provides configuration information for VMSAv8-64 and PMSAv8-64 virtualization using stage 2 of EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-144: AARCH64_VSCTLR_EL2 bit assignments

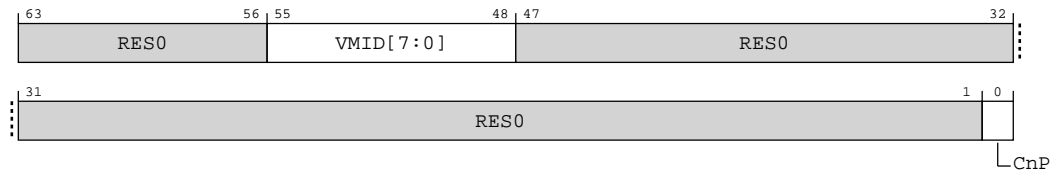


Table A-372: VSCTLR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:48]	VMID[7:0]	The VMID for the EL1-Guest-OS.	8 {x}
[47:1]	RES0	Reserved	RES0
[0]	CnP	<p>When AArch64-VTCR_EL2.MSA == '1'</p> <p>Common not Private. This bit indicates whether stage 2 of EL1&O translations are a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VSCTLR_EL2.CnP is 1.</p> <p>0b0</p> <p>The stage 2 translations of the EL1&O translation regime are permitted to differ in other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.</p> <p>0b1</p> <p>The stage 2 translations of the EL1&O translation regime are the same for every other PE in the Inner Shareable domain for which the value of VSCTLR_EL2.CnP is 1 and the VMID is the same as the current VMID.</p> <p>If the value of VSCTLR_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and the stage 2 EL1&O translation does not point to the same configurations when using the current VMID, then the results of the translations are <i>CONSTRAINED UNPREDICTABLE</i>, see <i>CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>In an implementation that does not support VMSAv8-64 at stage 1 EL1&O translation regime this field is RES0.</p> <p>Otherwise</p> <p>RES0</p>	xxxx

Access

MRS <Xt>, VSCTLR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b000

MSR VSCTLR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b000

Accessibility

MRS <Xt>, VSCTLR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = VSCTLR_EL2;
```

MSR VSCTLR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    VSCTLR_EL2 = X[t, 64];
```

A.2.2.85 TCR_EL2, Translation Control Register (EL2)

This register controls stage 1 of the EL2 translation regime, that supports a single VA range, translated using AArch64-TTBR0_EL2.

Configurations

This register is available in all configurations.

Attributes

Width

64

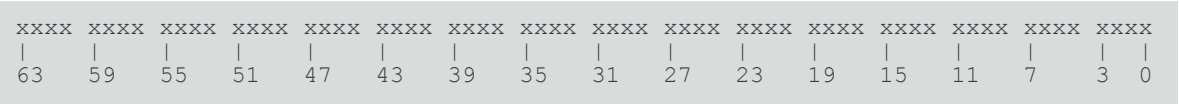
Functional group

Generic System Control

Access type

See bit descriptions

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

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

Figure A-145: AARCH64_TCR_EL2 bit assignments

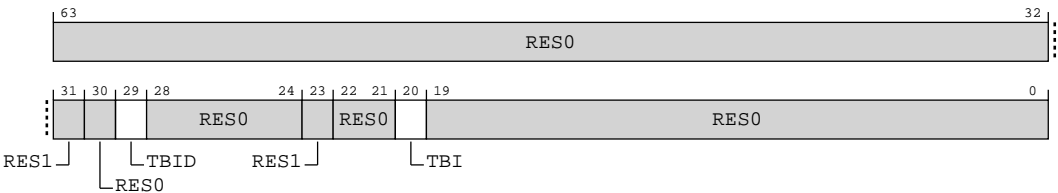


Table A-375: TCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	RES1	Reserved	RES1
[30]	RES0	Reserved	RES0
[29]	TBID	Controls the use of the top byte of instruction addresses for address matching. For the purpose of this field, all cache maintenance and address translation instructions that perform address translation are treated as data accesses. For more information, see <i>Address tagging</i> in the <i>Arm® Architecture Reference Manual for A-profile architecture</i> . 0b0 TCR_EL2.TBI applies to Instruction and Data accesses. 0b1 TCR_EL2.TBI applies to Data accesses only. This affects addresses where the address would be translated by EL2 MPU.	x
[28:24]	RES0	Reserved	RES0
[23]	RES1	Reserved	RES1
[22:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	TBI	<p>Top Byte ignored. Indicates whether the top byte of an address is used for address match for the EL2 MPU regions, or ignored and used for tagged addresses.</p> <p>For more information, see <i>Address tagging</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>0b0 Top Byte used in the address calculation.</p> <p>0b1 Top Byte ignored in the address calculation.</p> <p>This affects addresses generated in EL2 using AArch64 where the address would be translated by the EL2 MPU. It has an effect whether the EL2 translation regime is enabled or not.</p> <p>If FEAT_PAuth is implemented and TCR_EL2.TBID is 1, then this field only applies to Data accesses.</p> <p>If the value of TBI is 1, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:</p> <ul style="list-style-type: none"> • A branch or procedure return within EL2. • An exception taken to EL2. • An exception return to EL2. 	x
[19:0]	RES0	Reserved	RES0

Access

MRS <Xt>, TCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b010

MSR TCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0000	0b010

Accessibility

MRS <Xt>, TCR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TCR_EL2;

```

MSR TCR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then

```

```
TCR_EL2 = X[t, 64];
```

A.2.2.86 VTCR_EL2, Virtualization Translation Control Register

The control register for stage 2 of the EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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

Any of the bits in VTCR_EL2 are permitted to be cached in a TLB.

Figure A-146: AARCH64_VTCR_EL2 bit assignments

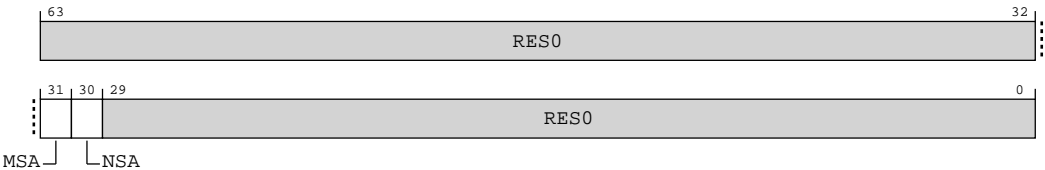


Table A-378: VTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	MSA	When AArch64-ID_AA64MMFR0_EL1.MSA_frac == '0010' Stage 1 EL1&0 translation regime memory system architecture. 0b0 Stage 1 EL1&0 translation regime uses PMSAv8-64 memory architecture. 0b1 Stage 1 EL1&0 translation regime uses VMSAv8-64 memory architecture. When AArch64-ID_AA64MMFR0_EL1.MSA_frac == '0001' RES1 Otherwise RES1	x
[30]	NSA	Non-secure stage 2 translation output address space. 0b0 All stage 2 translations for the Non-secure PA space of the Secure EL1&0 translation regime access the Secure PA space. 0b1 All stage 2 translations for the Non-secure PA space of the Secure EL1&0 translation regime access the Non-secure PA space. This bit behaves as 1 for all purposes other than reading back the value of the bit when the value of AArch64-VSTCR_EL2.SA is 1.	x
[29:0]	RES0	Reserved	RES0

Access

Unless stated otherwise, any of the bits in VTCR_EL2 are permitted to be cached in a TLB.

MRS <Xt>, VTCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b010

MSR VTCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0001	0b010

Accessibility

Unless stated otherwise, any of the bits in VTCR_EL2 are permitted to be cached in a TLB.

MRS <Xt>, VTCR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VTCR_EL2;

```


MSR VTCR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    VTCR_EL2 = X[t, 64];
```

A.2.2.87 VSTCR_EL2, Virtualization Secure Translation Control Register

The control register for stage 2 of the Secure EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

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



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Any of the bits in VSTCR_EL2 are permitted to be cached in a TLB.

Figure A-147: AARCH64_VSTCR_EL2 bit assignments

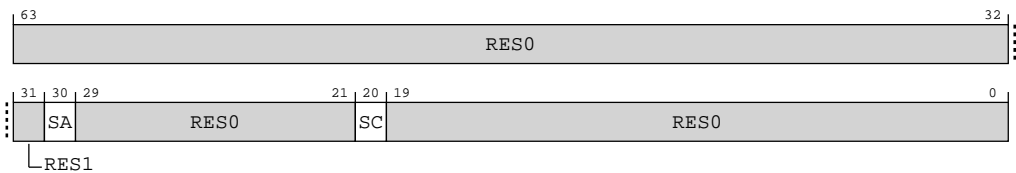


Table A-381: VSTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	RES1	Reserved	RES1
[30]	SA	Secure stage 2 translation output address space. 0b0 All stage 2 translations for the Secure PA space access the Secure PA space. 0b1 All stage 2 translations for the Secure PA space access the Non-secure PA space.	x
[29:21]	RES0	Reserved	RES0
[20]	SC	NS check enable bit. 0b0 Least secure NS configuration is selected from the stage 1 and stage 2 EL1&0 translation regime for the given address. 0b1 Stage 2 NS configuration is checked against stage 1 NS configuration in EL1&0 translation regime for the given address, and generate a fault if they are different.	x
[19:0]	RES0	Reserved	RES0

Access

MRS <Xt>, VSTCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0110	0b010

MSR VSTCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0010	0b0110	0b010

Accessibility

MRS <Xt>, VSTCR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = VSTCR_EL2;

```

MSR VSTCR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    VSTCR_EL2 = X[t, 64];

```

A.2.2.88 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)

Provides additional implementation defined fault status information for Data Abort, Instruction Abort or SError interrupt taken to EL2

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-148: AARCH64_AFSR0_EL2 bit assignments

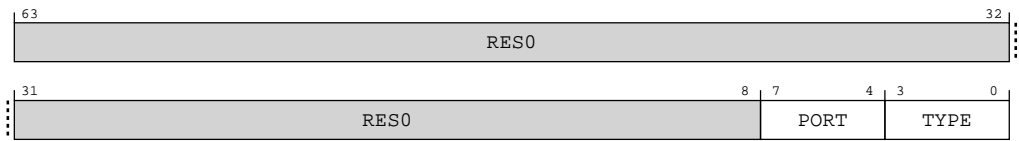


Table A-384: AFSR0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	PORT	<p>Memory or bus interface that caused the abort.</p> <p>0b0000 MM. Main Manager port.</p> <p>0b0001 LLRAM. Low-Latency RAM port.</p> <p>0b0010 LLPP. Low-Latency Peripheral Port.</p> <p>0b0011 SPP. Shared Peripheral Port.</p> <p>0b0100 ITCM. Instruction Tightly Coupled Memory.</p> <p>0b0101 DTCM. Data Tightly Coupled Memory.</p> <p>0b0110 Overlapping or illegal port. Used for accesses that target multiple ports, when they simultaneously belong in two or more of the ITCM, DTCM, LLRAM, LLPP or SPP enabled regions. Also used for page table accesses if they belong in any of the ITCM, DTCM, LLRAM, LLPP or SPP enabled regions.</p> <p>0b0111 Unknown. All faults other than Instruction or Data aborts will have this type. For External aborts and Unsupported Exclusive or Atomic access faults, this is used for ambiguous memory accesses or for faults that do not have an associated address. For other Instruction or Data aborts, this indicates that the port information is not applicable to the fault.</p> <p>Other values are Reserved.</p>	xxxx
[3:0]	TYPE	<p>Fault type.</p> <p>0b0000 Error on data.</p> <p>0b0010 Error on interrupt latency. Indicates the abort was generated by an instruction trapped by AArch64-IMP_INTLATENCY_EL2 controls.</p> <p>0b0011 Other error.</p> <p>Other values are Reserved.</p>	xxxx

Access

MRS <Xt>, AFSRO_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

MSR AFSRO_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0001	0b000

Accessibility

MRS <Xt>, AFSR0_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR0_EL2;
```

MSR AFSR0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR0_EL2 = X[t, 64];
```

A.2.2.89 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL2.

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-149: AARCH64_AFSR1_EL2 bit assignments

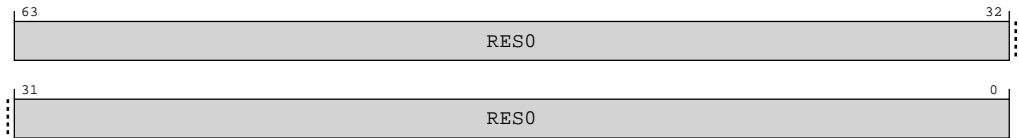


Table A-387: AFSR1_EL2 bit descriptions

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

Access

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

Accessibility

MRS <Xt>, AFSR1_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AFSR1_EL2;
```

MSR AFSR1_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR1_EL2 = X[t, 64];
```

A.2.2.90 ESR_EL2, Exception Syndrome Register (EL2)

Holds syndrome information for an exception taken to EL2.

Configurations

This register is available in all configurations.

Attributes

Width

64

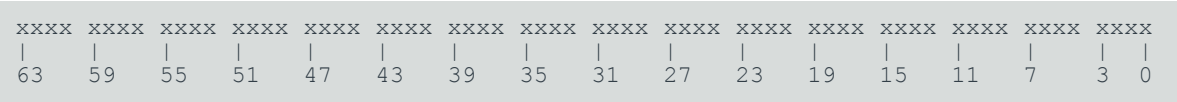
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

ESR_EL2 is made **UNKNOWN** as a result of an exception return from EL2.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL2, the value of ESR_EL2 is **UNKNOWN**. The value written to ESR_EL2 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

Figure A-150: AARCH64_ESR_EL2 bit assignments

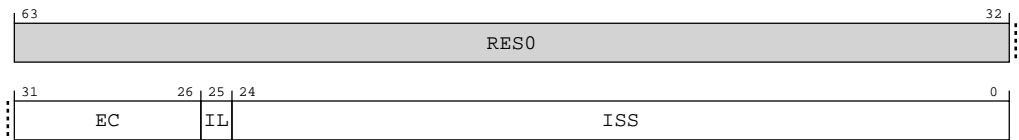


Table A-390: ESR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:26]	EC	<p>Exception Class. Indicates the reason for the exception that this register holds information about.</p> <p>For each EC value, the table references a subsection that gives information about:</p> <ul style="list-style-type: none"> The cause of the exception, for example the configuration required to enable the trap. The encoding of the associated ISS. <p>Possible values of the EC field are:</p>	6{x}
[31:26 continued]	EC	<p>0b000000 Unknown reason.</p> <p>0b000001 Trapped WFI or WFE instruction execution.</p> <p>Conditional WFE and WFI instructions that fail their condition code check do not cause an exception.</p> <p>0b000111 Access to Advanced SIMD, or floating-point functionality trapped by AArch64-CPACR_EL1.FPEN, AArch64-CPTR_EL2.FPEN, or AArch64-CPTR_EL2.TFP control.</p> <p>Excludes exceptions resulting from AArch64-CPACR_EL1 when the value of AArch64-HCR_EL2.TGE is 1, or because Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000.</p>	6{x}
[31:26 continued]	EC	<p>0b001001 Trapped use of a Pointer authentication instruction because HCR_EL2.API == 0.</p> <p>0b001110 Illegal Execution state.</p> <p>0b010101 SVC instruction execution in AArch64 state.</p> <p>0b010110 HVC instruction execution in AArch64 state, when HVC is not disabled.</p> <p>0b011000 Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111.</p> <p>This includes all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview', except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.</p>	6{x}
[31:26 continued]	EC	<p>0b011100 Exception from a Pointer Authentication instruction authentication failure</p> <p>0b100000 Instruction Abort from a lower Exception level.</p> <p>Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p> <p>0b100001 Instruction Abort taken without a change in Exception level.</p> <p>Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p>	6{x}

Bits	Name	Description	Reset
[31:26 continued]	EC	<p>0b100010 PC alignment fault exception.</p> <p>0b100100 Data Abort from a lower Exception level.</p> <p>Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p> <p>0b100101 Data Abort without a change in Exception level.</p> <p>Used for MMU faults generated by data accesses, alignment faults other than those caused by Stack Pointer misalignment, and synchronous External aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.</p> <p>0b100110 SP alignment fault exception.</p>	6{x}
[31:26 continued]	EC	<p>0b101111 SError interrupt.</p> <p>0b110000 Breakpoint exception from a lower Exception level.</p> <p>0b110001 Breakpoint exception taken without a change in Exception level.</p> <p>0b110010 Software Step exception from a lower Exception level.</p> <p>0b110011 Software Step exception taken without a change in Exception level.</p> <p>0b110100 Watchpoint from a lower Exception level.</p>	6{x}
[31:26 continued]	EC	<p>0b110101 Watchpoint exceptions without a change in Exception level.</p> <p>0b111100 BRK instruction execution in AArch64 state.</p> <p>All other EC values are reserved by Arm, and:</p> <ul style="list-style-type: none"> Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions. Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions. <p>The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE.</p>	6{x}

Bits	Name	Description	Reset
[25]	IL	<p>Instruction Length for synchronous exceptions. Possible values of this bit are:</p> <p>0b0</p> <p>16-bit instruction trapped.</p> <p>0b1</p> <p>32-bit instruction trapped. This value is also used when the exception is one of the following:</p> <ul style="list-style-type: none"> • An SError interrupt. • An Instruction Abort exception. • A PC alignment fault exception. • An SP alignment fault exception. • A Data Abort exception for which the value of the ISV bit is 0. • An Illegal Execution state exception. • Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning: <ul style="list-style-type: none"> ◦ 0b1: 32-bit A64 BRK instruction. • An exception reported using EC value 0b000000. 	x
[24:0]	ISS	<p>ISS encoding for exceptions with an unknown reason</p> <p>24:0</p> <p>Reserved, RES0.</p> <p>When an exception is reported using this EC code the IL field is set to 1.</p> <p>This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:</p>	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> • The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including: <ul style="list-style-type: none"> ◦ A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> • A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state. • Instruction encodings that are unallocated. • Instruction encodings for instructions or System registers that are not implemented in the implementation. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> • In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state. • In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> • In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> • Attempted execution of: <ul style="list-style-type: none"> ◦ An HVC instruction when disabled by AArch64-HCR_EL2.HCD. ◦ An SMC instruction. ◦ An HLT instruction when disabled by ext-EDSCR.HDE. 	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	<ul style="list-style-type: none"> Attempted execution of an MSR or MRS instruction to access AArch64-SP_ELO when the value of AArch64-SPSel.SP is 0. Attempted execution of an MSR or MRS instruction using a _EL12 register name. 	25 {x}
[24:0 continued]	ISS	Attempted execution, in Debug state, of: <ul style="list-style-type: none"> A DCPS1 instruction when the value of AArch64-HCR_EL2.TGE is 1 and EL2 is disabled or not implemented in the current Security state. A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state. A DCPS3 instruction. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> An exception that is taken to EL2 because the value of AArch64-HCR_EL2.TGE is 1 that, if the value of AArch64-HCR_EL2.TGE was 0 would have been reported with an ESR_ELx.EC value of 0b000111. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from a WF* instruction 24 Condition code valid. 0b0 The COND field is not valid. 0b1 The COND field is valid.	25 {x}
[24:0 continued]	ISS	23:20 For exceptions taken from AArch64, this field is set to 0b1110. 19:2 Reserved, RES0. 1:0 Trapped instruction. Possible values of this bit are: 0b00 WFI trapped. 0b01 WFE trapped.	25 {x}
[24:0 continued]	ISS	The following fields describe configuration settings for generating this exception: <ul style="list-style-type: none"> AArch64-SCTLR_EL1.{nTWE, nTWI}. AArch64-HCR_EL2.{TWE, TWI}. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps The accesses covered by this trap include: <ul style="list-style-type: none"> Execution of SVE or Advanced SIMD and floating-point instructions. Accesses to the Advanced SIMD and floating-point System registers. 	25 {x}
[24:0 continued]	ISS	For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value 0b000000.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	24 Condition code valid. 0b0 The COND field is not valid. 0b1 The COND field is valid.	25 {x}
[24:0 continued]	ISS	23:20 For exceptions taken from AArch64, this field is set to 0b1110. 19:0 Reserved, RES0 .	25 {x}
[24:0 continued]	ISS	The following fields describe the configuration settings for the traps that are reported using EC value 0b000111: <ul style="list-style-type: none"> AArch64-CPACR_EL1.FPEN, for accesses to SIMD and floating-point registers trapped to EL1. AArch64-CPTR_EL2.FPEN and AArch64-CPTR_EL2.TFP, for accesses to SIMD and floating-point registers trapped to EL2. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault 24:0 Reserved, RES0 . There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about PC alignment fault exceptions, see <i>PC alignment checking</i> in the Arm® Architecture Reference Manual for A-profile architecture . <i>SP alignment checking</i> in the Arm® Architecture Reference Manual for A-profile architecture describes the configuration settings for generating SP alignment fault exceptions.	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from HVC or SVC instruction execution 24:16 Reserved, RES0 . For A64 instructions, see SVC in the and 'HVC'.	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state 24:22 Reserved, RES0 . 21:20 The Op0 value from the issued instruction. 19:17 The Op2 value from the issued instruction.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	16:14 The Op1 value from the issued instruction. 13:10 The CRn value from the issued instruction. 9:5 The Rt value from the issued instruction, the general-purpose register used for the transfer.	25 {x}
[24:0 continued]	ISS	4:1 The CRm value from the issued instruction. 0 Indicates the direction of the trapped instruction. 0b0 Write access, including MSR instructions. 0b1 Read access, including MRS instructions.	25 {x}
[24:0 continued]	ISS	For exceptions caused by System instructions, see <i>System instructions' subsection of 'Branches, exception generating and System instructions</i> in the Arm® Architecture Reference Manual for A-profile architecture for the encoding values returned by an instruction. The following fields describe configuration settings for generating the exception that is reported using EC value 0b011000:	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-SCTLR_EL1.UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-SCTLR_EL1.UCT, for accesses to AArch64-CTR_EL0 using AArch64 state, MSR or MRS access trapped to EL1 or EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-SCTLR_EL1.DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-SCTLR_EL1.UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-CPACR_EL1.TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-CNTKCTL_EL1.{ELOPTEN, EL0VTEN, EL0PCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. AArch64-PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-HCR_EL2.{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.TACR, for accesses to the Auxiliary Control Register, AArch64-ACTLR_EL1, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-HCR_EL2.{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-CPTR_EL2.TCPAC, for accesses to AArch64-CPACR_EL1, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-CPTR_EL2.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-MDCR_EL2.TTRF, for accesses to the trace filter control register, AArch64-TRFCR_EL1, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-MDCR_EL2.TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2. AArch64-MDCR_EL2.TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2. AArch64-MDCR_EL2.TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2. AArch64-MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> AArch64-HCR_EL2.APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2. AArch64-HCR_EL2.AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2. AArch64-HCR_EL2.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from an Instruction Abort 24:13 Reserved, RES0.	25 {x}
[24:0 continued]	ISS	12:11 Synchronous Error Type. When IFSC is 0b010000, describes the PE error state after taking the Instruction Abort exception. 0b00 Recoverable state (UER). 0b10 Uncontainable (UC). 0b11 Restartable state (UEO).	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	10 FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk. 0b0 FAR is valid. 0b1 FAR is not valid, and holds an UNKNOWN value.	25 {x}
[24:0 continued]	ISS	9 External abort type. 0b0 No IMPLEMENTATION DEFINED classification of External aborts.	25 {x}
[24:0 continued]	ISS	8 Reserved, RES0 . 7 For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:	25 {x}
[24:0 continued]	ISS	0b0 Fault not on a stage 2 translation for a stage 1 translation table walk. 0b1 Fault on the stage 2 translation of an access for a stage 1 translation table walk.	25 {x}
[24:0 continued]	ISS	6 Reserved, RES0 .	25 {x}
[24:0 continued]	ISS	5:0 Instruction Fault Status Code. 0b000000 Address size fault, level 0 of translation or translation table base register. 0b000001 Address size fault, level 1. 0b000010 Address size fault, level 2.	25 {x}
[24:0 continued]	ISS	0b000011 Address size fault, level 3. 0b000100 Translation fault, level 0. 0b000101 Translation fault, level 1. 0b000110 Translation fault, level 2.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	0b000111 Translation fault, level 3. 0b001001 Access flag fault, level 1. 0b001010 Access flag fault, level 2. 0b001011 Access flag fault, level 3. 0b001100 Permission fault, level 0.	25 {x}
[24:0 continued]	ISS	0b001101 Permission fault, level 1. 0b001110 Permission fault, level 2. 0b001111 Permission fault, level 3. 0b010000 Synchronous External abort, not on translation table walk or hardware update of translation table.	25 {x}
[24:0 continued]	ISS	0b010100 Synchronous External abort on translation table walk or hardware update of translation table, level 0. 0b010101 Synchronous External abort on translation table walk or hardware update of translation table, level 1.	25 {x}
[24:0 continued]	ISS	0b010110 Synchronous External abort on translation table walk or hardware update of translation table, level 2. 0b010111 Synchronous External abort on translation table walk or hardware update of translation table, level 3. 0b110000 TLB conflict abort. 0b110001 Unsupported atomic hardware update fault.	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from a Data Abort 24 Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid. 0b0 No valid instruction syndrome. ISS[23:14] are RES0. 0b1 ISS[23:14] hold a valid instruction syndrome. 13 Reserved, RES0.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	10 FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk. 0b0 FAR is valid. 0b1 FAR is not valid, and holds an UNKNOWN value.	25 {x}
[24:0 continued]	ISS	9 External abort type. 0b0 No IMPLEMENTATION DEFINED classification of External aborts.	25 {x}
[24:0 continued]	ISS	8 Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction: 0b0 The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.	25 {x}
[24:0 continued]	ISS	0b1 The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The r[DC ZVA](AArch64-dc-zva), r[DC GVA](AArch64-dc-gva), and r[DC GZVA](AArch64-dc-gzva) instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.	25 {x}
[24:0 continued]	ISS	7 For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk: 0b0 Fault not on a stage 2 translation for a stage 1 translation table walk. 0b1 Fault on the stage 2 translation of an access for a stage 1 translation table walk.	25 {x}
[24:0 continued]	ISS	6 Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location. 0b0 Abort caused by an instruction reading from a memory location. 0b1 Abort caused by an instruction writing to a memory location.	25 {x}
[24:0 continued]	ISS	5:0 Data Fault Status Code. 0b000000 Address size fault, level 0 of translation or translation table base register.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	0b000001 Address size fault, level 1. 0b000010 Address size fault, level 2. 0b000011 Address size fault, level 3. 0b000100 Translation fault, level 0.	25 {x}
[24:0 continued]	ISS	0b000101 Translation fault, level 1. 0b000110 Translation fault, level 2. 0b000111 Translation fault, level 3. 0b001001 Access flag fault, level 1.	25 {x}
[24:0 continued]	ISS	0b001010 Access flag fault, level 2. 0b001011 Access flag fault, level 3. 0b001100 Permission fault, level 0. 0b001101 Permission fault, level 1.	25 {x}
[24:0 continued]	ISS	0b001110 Permission fault, level 2. 0b001111 Permission fault, level 3. 0b010000 Synchronous External abort, not on translation table walk or hardware update of translation table. 0b010100 Synchronous External abort on translation table walk or hardware update of translation table, level 0.	25 {x}
[24:0 continued]	ISS	0b010101 Synchronous External abort on translation table walk or hardware update of translation table, level 1. 0b010110 Synchronous External abort on translation table walk or hardware update of translation table, level 2. 0b010111 Synchronous External abort on translation table walk or hardware update of translation table, level 3. 0b100001 Alignment fault.	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	0b110000 TLB conflict abort. 0b110001 Unsupported atomic hardware update fault. 0b110100 IMPLEMENTATION DEFINED fault (Lockdown). 0b110101 IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access).	25 {x}
[24:0 continued]	ISS	ISS encoding for an SError interrupt 24 IMPLEMENTATION DEFINED syndrome. 0b0 Bits [23:0] of the ISS field holds the fields described in this encoding	25 {x}
[24:0 continued]	ISS	0b1 Bits [23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.	25 {x}
[24:0 continued]	ISS	23:14 Reserved, RES0 .	25 {x}
[24:0 continued]	ISS	13 Implicit error synchronization event. 0b0 The SError interrupt was either not synchronized by the implicit error synchronization event or not taken immediately. 0b1 The SError interrupt was synchronized by the implicit error synchronization event and taken immediately.	25 {x}
[24:0 continued]	ISS	12:10 Asynchronous Error Type. When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception. 0b000 Uncontainable (UC). 0b001 Unrecoverable state (UEU). 0b010 Restartable state (UEO). 0b011 Recoverable state (UER). 0b110 Corrected (CE).	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	<p>9</p> <p>External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts.</p> <p>This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.</p> <p>8:6</p> <p>Reserved, RES0.</p> <p>5:0</p> <p>Data Fault Status Code.</p> <p>0b000000</p> <p>Uncategorized error.</p> <p>0b010001</p> <p>Asynchronous SError interrupt.</p>	25 {x}
[24:0 continued]	ISS	<p>ISS encoding for an exception from a Breakpoint or Vector Catch debug exception</p> <p>24:6</p> <p>Reserved, RES0.</p> <p>5:0</p> <p>Instruction Fault Status Code.</p> <p>0b100010</p> <p>Debug exception.</p> <p>For more information about generating these exceptions:</p> <ul style="list-style-type: none"> For exceptions from AArch64, see <i>Breakpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture. 	25 {x}
[24:0 continued]	ISS	<p>ISS encoding for an exception from a Software Step exception</p> <p>24</p> <p>Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:</p> <p>0b0</p> <p>EX bit is RES0.</p> <p>0b1</p> <p>EX bit is valid.</p> <p>23:7</p> <p>Reserved, RES0.</p>	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	<p>6</p> <p>Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.</p> <p>0b0</p> <p>An instruction other than a Load-Exclusive instruction was stepped.</p> <p>0b1</p> <p>A Load-Exclusive instruction was stepped.</p> <p>5:0</p> <p>Instruction Fault Status Code.</p> <p>0b100010</p> <p>Debug exception.</p> <p>For more information about generating these exceptions, see <i>Software Step exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	25 {x}
[24:0 continued]	ISS	<p>ISS encoding for an exception from a Watchpoint exception</p> <p>24:9</p> <p>Reserved, RES0.</p> <p>8</p> <p>Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation instruction:</p>	25 {x}
[24:0 continued]	ISS	<p>0b0</p> <p>The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1.</p> <p>0b1</p> <p>The Watchpoint exception was generated by either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation instruction. The r[DC ZVA](AArch64-dc-zva), r[DC GVA](AArch64-dc-gva), and r[DC GZVA](AArch64-dc-gzva) instructions are not classified as a cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.</p>	25 {x}
[24:0 continued]	ISS	<p>7</p> <p>Reserved, RES0.</p>	25 {x}
[24:0 continued]	ISS	<p>6</p> <p>Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.</p> <p>0b0</p> <p>Watchpoint exception caused by an instruction reading from a memory location.</p> <p>0b1</p> <p>Watchpoint exception caused by an instruction writing to a memory location</p>	25 {x}
[24:0 continued]	ISS	<p>5:0</p> <p>Data Fault Status Code.</p> <p>0b100010</p> <p>Debug exception.</p> <p>For more information about generating these exceptions, see <i>Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	25 {x}

Bits	Name	Description	Reset
[24:0 continued]	ISS	ISS encoding for an exception from execution of a Breakpoint instruction 24:16 Reserved, RES0. 15:0 Set to the instruction comment field value, zero extended as necessary. For more information about generating these exceptions, see <i>Breakpoint instruction exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture .	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from a Pointer Authentication instruction when HCR_EL2.API == 0 SCR_EL3.API == 0 24:0 Reserved, RES0. For more information about generating these exceptions, see: <ul style="list-style-type: none"> AArch64-HCR_EL2.API, for exceptions from Pointer authentication instructions, using AArch64 state, trapped to EL2. 	25 {x}
[24:0 continued]	ISS	ISS encoding for an exception from a Pointer Authentication instruction authentication failure 24:2 Reserved, RES0.	25 {x}
[24:0 continued]	ISS	1 This field indicates whether the exception is as a result of an Instruction key or a Data key. 0b0 Instruction Key. 0b1 Data Key. 0 This field indicates whether the exception is as a result of an A key or a B key. 0b0 A key. 0b1 B key	25 {x}
[24:0 continued]	ISS	The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect: <ul style="list-style-type: none"> AUTIASP, AUTIAZ, AUTIA1716. AUTIBSP, AUTIBZ, AUTIB1716. AUTIA, AUTDA, AUTIB, AUTDB. AUTIZA, AUTIZB, AUTDZA, AUTDZB. 	25 {x}
[24:0 continued]	ISS	<ul style="list-style-type: none"> RETAA, RETAB. BRAA, BRAB, BLRAA, BLRAB. BRAAZ, BRABZ, BLRAAZ, BLRABZ. ERETAA, ERETAB. LDRAA, LDRAB, whether the authenticated address is written back to the base register or not. 	25 {x}

Access

MRS <Xt>, ESR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

MSR ESR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

Accessibility

MRS <Xt>, ESR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ESR_EL2;
```

MSR ESR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    ESR_EL2 = X[t, 64];
```

A.2.2.91 PRBAR<n>_EL2, Protection Region Base Address Register n (EL2), n = 1 - 15

Provides access to the base address for the MPU region determined by the value of 'n' and AArch64-PRSELR_EL2.REGION as AArch64-PRSELR_EL2.REGION<7:4>:n.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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_PRBAR<n>_EL2 bit assignments

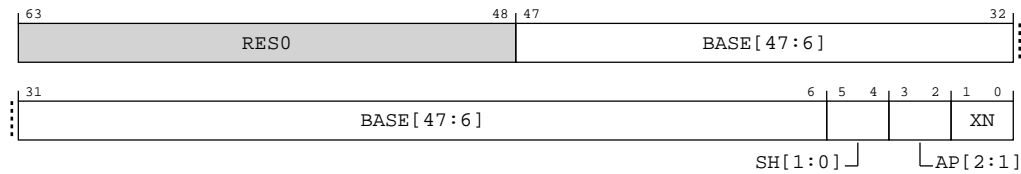


Table A-393: PRBAR<n>_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:6]	BASE[47:6]	Bits[47:6] of the lower inclusive limit of the selected EL2 MPU memory region. This value is zero extended to provide the base address to be checked against.	42 {x}
[5:4]	SH[1:0]	Shareability attribute. 0b00 Non-shareable 0b01 Reserved, CONSTRAINED UNPREDICTABLE 0b10 Outer Shareable 0b11 Inner Shareable	xx
[3:2]	AP[2:1]	Access Permission attributes. 0b00 Read/write at EL2, no access at EL1 or ELO 0b01 Read/write at EL2, EL1 and ELO 0b10 Read-only at EL2, no access at EL1 or ELO 0b11 Read-only at EL2, EL1 and ELO	xx

Bits	Name	Description	Reset
[1:0]	XN	<p>Execute Never. For</p> <ul style="list-style-type: none"> Stage 1 EL2 translation regime and Stage 2 EL1&0 translation regime when FEAT_XNX is not implemented <p>XN[1] determines whether execution of the instructions fetched from the MPU memory region is permitted. In this case, XN[0] is RES0</p> <p>For stage 2 EL1&0 translation regime when FEAT_XNX is implemented, the behavior of XN[1:0] is same as that defined by VMSAv8-64 for EL1&0 stage 2 translation table XN[1:0].bits[54:53] field in Armv8-A architecture.</p> <p>0b00 Execution of instructions fetched from the region is permitted.</p> <p>0b01 Execution of instructions fetched from the region is not permitted.</p>	xx

Access

Any access to MPU region register PRBAR<n>_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR<n>_EL2 return an **UNKNOWN** value.
- Writes to unimplemented PRBAR<n>_EL2 register make all PRBAR_EL2 registers value **UNKNOWN**.

MRS <Xt>, PRBAR<m>_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	'1':m[3:1]	m[0]:'00'

MSR PRBAR<m>_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	'1':m[3:1]	m[0]:'00'

Accessibility

Any access to MPU region register PRBAR<n>_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR<n>_EL2 return an **UNKNOWN** value.
- Writes to unimplemented PRBAR<n>_EL2 register make all PRBAR_EL2 registers value **UNKNOWN**.

MRS <Xt>, PRBAR<m>_EL2

```
integer m = UInt(CRm<2:0>:op2<2>);
```

```

if m + (UInt(PRSELR_EL2.REGION<7:4>) * 16) >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PRBAR_EL2[m + (UInt(AArch64-PRSELR_EL2.REGION<7:4>) * 16)];

```

MSR PRBAR<m>_EL2, <Xt>

```

integer m = UInt(CRm<2:0>:op2<2>);
if m + (UInt(PRSELR_EL2.REGION<7:4>) * 16) >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    PRBAR_EL2[m + (UInt(AArch64-PRSELR_EL2.REGION<7:4>) * 16)] = X[t, 64];

```

A.2.2.92 PRLAR<n>_EL2, Protection Region Limit Address Register n (EL2), n = 1 - 15

Provides access to the limit address for the MPU region determined by the value of 'n' and AArch64-PRSELR_EL2.REGION as AArch64-PRSELR_EL2.REGION<7:4>:n.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

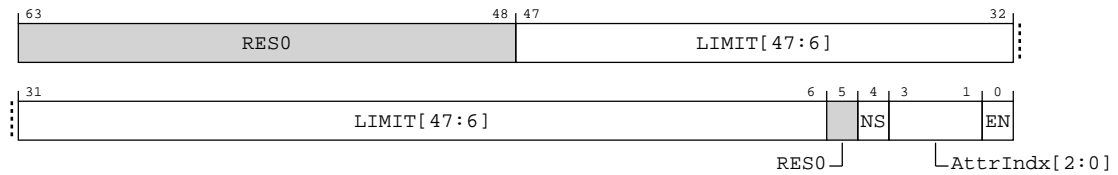
Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-152: AARCH64_PRLAR<n>_EL2 bit assignments**Table A-396: PRLAR<n>_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:6]	LIMIT[47:6]	Bits[47:6] of the upper inclusive limit of the selected EL2 MPU memory region. This value is concatenated with the value 0x3F to provide the limit address to be checked against.	42 {x}
[5]	RES0	Reserved	RES0
[4]	NS	Non-secure bit. Specifies whether the output address is in the Secure or Non-secure memory. 0b0 Output address is in Secure address space. 0b1 Output address is in Non-secure address space.	x
[3:1]	AttrIdx[2:0]	Selects attributes from within the associated Memory Attribute Indirection Register. 0b000 Select the Attr0 field from MAIR_EL2. 0b001 Select the Attr1 field from MAIR_EL2. 0b010 Select the Attr2 field from MAIR_EL2. 0b011 Select the Attr3 field from MAIR_EL2. 0b100 Select the Attr4 field from MAIR_EL2. 0b101 Select the Attr5 field from MAIR_EL2. 0b110 Select the Attr6 field from MAIR_EL2. 0b111 Select the Attr7 field from MAIR_EL2.	xxx

Bits	Name	Description	Reset
[0]	EN	Region enable bit. 0b0 Region disabled. 0b1 Region enabled.	0b0

Access

Any access to MPU region register PRLAR<n>_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR<n>_EL2 return an **UNKNOWN** value.
- Writes to unimplemented PRLAR<n>_EL2 register make all PRLAR_EL2 registers value **UNKNOWN**.

MRS <Xt>, PRLAR<m>_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	'1':m[3:1]	m[0]:'01'

MSR PRLAR<m>_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	'1':m[3:1]	m[0]:'01'

Accessibility

Any access to MPU region register PRLAR<n>_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR<n>_EL2 return an **UNKNOWN** value.
- Writes to unimplemented PRLAR<n>_EL2 register make all PRLAR_EL2 registers value **UNKNOWN**.

MRS <Xt>, PRLAR<m>_EL2

```
integer m = UInt(CRm<2:0>:op2<2>);
if (UInt(PRSELR_EL2.REGION<7:4>) * 16) + m >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRLAR_EL2[m + (UInt(AArch64-PRSELR_EL2.REGION<7:4>) * 16)];
```

MSR PRLAR<m>_EL2, <Xt>

```
integer m = UInt(CRm<2:0>:op2<2>);
if (UInt(PRSELR_EL2.REGION<7:4>) * 16) + m >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    PRLAR_EL2[m + (UInt(AArch64-PRSELR_EL2.REGION<7:4>) * 16)] = X[t, 64];
```

A.2.2.93 FAR_EL2, Fault Address Register (EL2)

Holds the faulting Virtual Address for all synchronous Instruction Abort exceptions, Data Abort exceptions, PC alignment fault exceptions and Watchpoint exceptions that are taken to EL2.

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-153: AARCH64_FAR_EL2 bit assignments

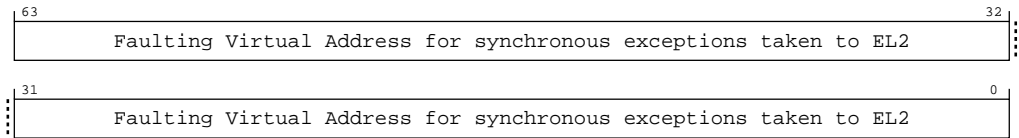


Table A-399: FAR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	<p>Faulting Virtual Address for synchronous exceptions taken to EL2. Exceptions that set the FAR_EL2 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), PC alignment faults (EC 0x22), and Watchpoints (EC 0x34 or 0x35). AArch64-ESR_EL2.EC holds the EC value for the exception.</p> <p>For a synchronous External abort, if the VA that generated the abort was from an address range for which TCR_ELx.TBI{<0 1>} == 1 for the translation regime in use when the abort was generated, then the top eight bits of FAR_EL2 are UNKNOWN.</p> <p>For a synchronous External abort other than a synchronous External abort on a translation table walk, this field is valid only if AArch64-ESR_EL2.FnV is 0, and FAR_EL2 is UNKNOWN if AArch64-ESR_EL2.FnV is 1.</p> <p>If a memory fault that sets FAR_EL2, other than a Tag Check Fault, is generated from a data cache maintenance or other DC instruction, this field holds the address specified in the register argument of the instruction.</p> <p>For a Data Abort exception or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see <i>Address tagging</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>Execution at EL1 or EL0 makes FAR_EL2 become UNKNOWN.</p> <p>Note: The address held in this field is an address accessed by the instruction fetch or data access that caused the exception that actually gave rise to the instruction or data abort. It is the lower address that gave rise to the fault that is reported. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize which fault is reported.</p> <p>For all other exceptions taken to EL2, FAR_EL2 is UNKNOWN.</p> <p>FAR_EL2 is made UNKNOWN on an exception return from EL2.</p>	64 {x}

Access

MRS <Xt>, FAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

MSR FAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

Accessibility

MRS <Xt>, FAR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = FAR_EL2;
```

MSR FAR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    FAR_EL2 = X[t, 64];
```

A.2.2.94 HPFAR_EL2, Hypervisor IPA Fault Address Register


Holds the faulting IPA for some aborts on a stage 2 translation taken to EL2.

Configurations

The HPFAR_EL2 is written for:

- Translation or Access faults in the second stage of translation.
- An abort in the second stage of translation performed during the translation table walk of a first stage translation, caused by a Translation fault, an Access flag fault, or a Permission fault.
- A stage 2 Address size fault.

For all other exceptions taken to EL2, this register is UNKNOWN.



Note

The address held in this register is an address accessed by the instruction fetch or data access that caused the exception that gave rise to the Instruction Abort exception or Data Abort exception. It is the lower address that gave rise to the fault that is reported. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize which fault is reported.

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Execution at EL1 or EL0 makes HPFAR_EL2 become **UNKNOWN**.

Figure A-154: AARCH64_HPFAAR_EL2 bit assignments

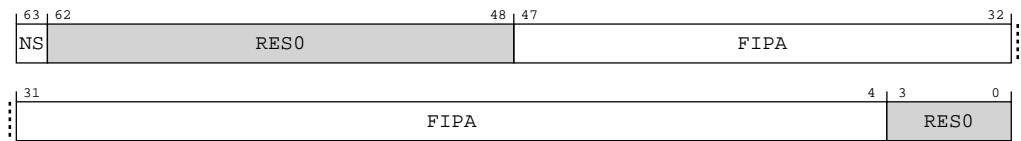


Table A-402: HPFAR_EL2 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Faulting IPA address space. 0b0 Faulting IPA is from the Secure IPA space. 0b1 Faulting IPA is from the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
47:4	FIPA	FIPA encoding for None 43:36 Reserved, RES0. 35:0 Bits[47:12] Faulting Intermediate Physical Address. For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.	4 4 { x }
[3:0]	RES0	Reserved	RES0

Access

MRS <Xt>, HPFAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b100

MSR HPFAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b100

Accessibility

MRS <Xt>, HPFAR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = HPFAR_EL2;
```

MSR HPFAR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    HPFAR_EL2 = X[t, 64];
```

A.2.2.95 PRENR_EL2, Protection Region Enable Register (EL2)

Provides direct access to the AArch64-PRLAR_EL2.EN bits of EL2 MPU regions from 0 to 31.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

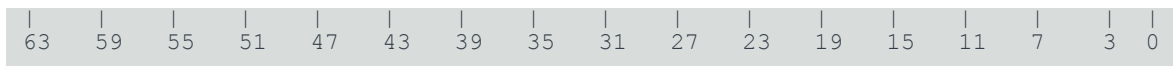
Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000	0000	0000	0000	0000	0000	0000
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-155: AARCH64_PRENR_EL2 bit assignments

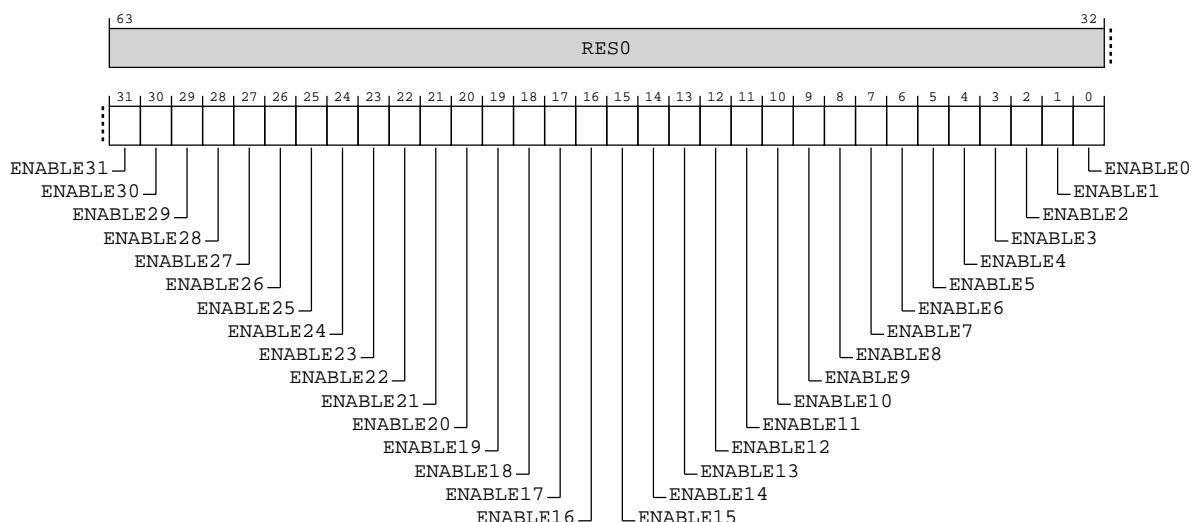


Table A-405: PRENR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ENABLE<n>, bit[n], where n = 31 to 0	<p>Enable bit. Each bit, n, enables or disables the respective EL2 MPU region. The bits associated with the unimplemented MPU regions are RAZ/WI.</p> <p>0b0</p> <p>Disables the EL2 MPU n region.</p> <p>0b1</p> <p>Enables the EL2 MPU n region.</p>	0x00000000

Access

MRS <Xt>, PRENR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0001	0b001

MSR PRENR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0001	0b001

Accessibility

MRS <Xt>, PRENR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRENR_EL2;
```

MSR PRENR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    PRENR_EL2 = X[t, 64];
```

A.2.2.96 PRSELR_EL2, Protection Region Selection Register (EL2)

Selects the region number for the EL2 MPU region associated with the AArch64-PRBAR_EL2 and AArch64-PRLAR_EL2 registers.

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-156: AARCH64_PRSELR_EL2 bit assignments

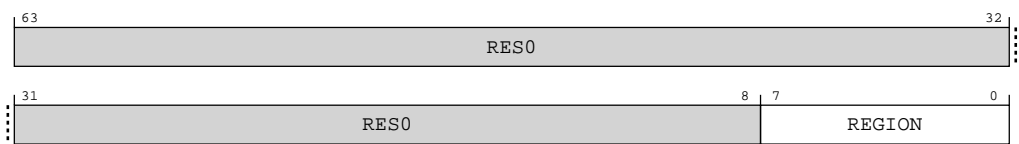


Table A-408: PRSELR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	REGION	The number of the current EL2 MPU region visible in AArch64-PRBAR_EL2 and AArch64-PRLAR_EL2. For N implemented MPU regions, memory region numbering starts at 0 and increments by 1 to the value N-1. Writing a value greater than or equal to the number of implemented MPU regions specified by AArch64-MPUIR_EL2.REGION, results in CONSTRAINED UNPREDICTABLE behavior. CONSTRAINED UNPREDICTABLE behavior is that PRSELR_EL2 register becomes UNKNOWN .	8{x}

Access

MRS <Xt>, PRSELR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0010	0b001

MSR PRSELR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0010	0b001

Accessibility

MRS <Xt>, PRSELR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRSELR_EL2;
```

MSR PRSELR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    PRSELR_EL2 = X[t, 64];
```

A.2.2.97 PRBAR_EL2, Protection Region Base Address Register (EL2)

Provides access to the base addresses for the EL2 MPU region. AArch64-PRSELR_EL2.REGION determines which MPU region is selected.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

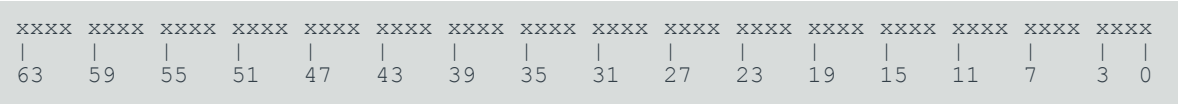
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-157: AARCH64_PRBAR_EL2 bit assignments

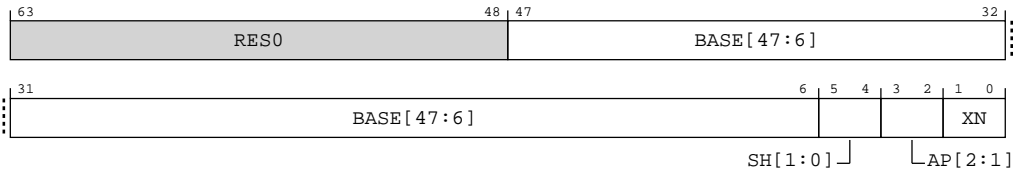


Table A-411: PRBAR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:6]	BASE[47:6]	Bits[47:6] of the lower inclusive limit of the selected EL2 MPU memory region. This value is zero extended to provide the base address to be checked against.	42 {x}

Bits	Name	Description	Reset
[5:4]	SH[1:0]	Shareability attribute. 0b00 Non-shareable 0b01 Reserved, CONSTRAINED UNPREDICTABLE 0b10 Outer Shareable 0b11 Inner Shareable	xx
[3:2]	AP[2:1]	Access Permission attributes. 0b00 Read/write at EL2, no access at EL1 or EL0 0b01 Read/write at EL2, EL1 and EL0 0b10 Read-only at EL2, no access at EL1 or EL0 0b11 Read-only at EL2, EL1 and EL0	xx
[1:0]	XN	Execute Never. For <ul style="list-style-type: none"> Stage 1 EL2 translation regime and Stage 2 EL1&0 translation regime when FEAT_XNX is not implemented <p>XN[1] determines whether execution of the instructions fetched from the MPU memory region is permitted. In this case, XN[0] is RES0</p> <p>For stage 2 EL1&0 translation regime when FEAT_XNX is implemented, the behavior of XN[1:0] is same as that defined by VMSAv8-64 for EL1&0 stage 2 translation table XN[1:0],bits[54:53] field in Armv8-A architecture.</p> 0b00 Execution of instructions fetched from the region is permitted. 0b01 Execution of instructions fetched from the region is not permitted.	xx

Access

Any access to MPU region register PRBAR_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR_EL2 register return an **UNKNOWN** value.
- Writes to unimplemented PRBAR_EL2 register make all PRBAR_EL2 registers value **UNKNOWN**.

MRS <Xt>, PRBAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b1000	0b000

MSR PRBAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b1000	0b000

Accessibility

Any access to MPU region register PRBAR_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is CONSTRAINED UNPREDICTABLE.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRBAR_EL2 register return an UNKNOWN value.
- Writes to unimplemented PRBAR_EL2 register make all PRBAR_EL2 registers value UNKNOWN.

MRS <Xt>, PRBAR_EL2

```

if UInt(PRSELR_EL2.REGION) >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRBAR_EL2[UInt(AArch64-PRSELR_EL2.REGION)];

```

MSR PRBAR_EL2, <Xt>

```

if UInt(PRSELR_EL2.REGION) >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    PRBAR_EL2[UInt(AArch64-PRSELR_EL2.REGION)] = X[t, 64];

```

A.2.2.98 PRLAR_EL2, Protection Region Limit Address Register (EL2)

Provides access to the limit addresses for the EL2 MPU region. AArch64-PRSELR_EL2.REGION determines which MPU region is selected.

Configurations

All bits above implemented physical address range in this register should be treated as RES0.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-158: AARCH64_PRLAR_EL2 bit assignments

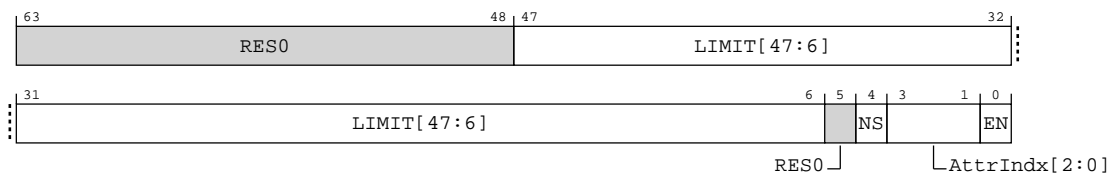


Table A-414: PRLAR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:6]	LIMIT[47:6]	Bits[47:6] of the upper inclusive limit of the selected EL2 MPU memory region. This value is concatenated with the value 0x3F to provide the limit address to be checked against.	42 {x}
[5]	RES0	Reserved	RES0
[4]	NS	Non-secure bit. Specifies whether the output address is in the Secure or Non-secure memory. 0b0 Output address is in Secure address space. 0b1 Output address is in Non-secure address space.	x

Bits	Name	Description	Reset
[3:1]	AttrIdx[2:0]	<p>Selects attributes from within the associated Memory Attribute Indirection Register.</p> <p>0b000 Select the Attr0 field from MAIR_EL2.</p> <p>0b001 Select the Attr1 field from MAIR_EL2.</p> <p>0b010 Select the Attr2 field from MAIR_EL2.</p> <p>0b011 Select the Attr3 field from MAIR_EL2.</p> <p>0b100 Select the Attr4 field from MAIR_EL2.</p> <p>0b101 Select the Attr5 field from MAIR_EL2.</p> <p>0b110 Select the Attr6 field from MAIR_EL2.</p> <p>0b111 Select the Attr7 field from MAIR_EL2.</p>	xxx
[0]	EN	<p>Region enable bit.</p> <p>0b0 Region disabled.</p> <p>0b1 Region enabled.</p>	0b0

Access

Any access to MPU region register PRLAR_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is **CONSTRAINED UNPREDICTABLE**.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR_EL2 register return an **UNKNOWN** value.
- Writes to unimplemented PRLAR_EL2 register make all PRLAR_EL2 registers value **UNKNOWN**.

MRS <Xt>, PRLAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b1000	0b001

MSR PRLAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b1000	0b001

Accessibility

Any access to MPU region register PRLAR_EL2 above the number of implemented regions specified by AArch64-MPUIR_EL2.REGION is CONSTRAINED UNPREDICTABLE.

CONSTRAINED UNPREDICTABLE behavior is defined as:

- Reads of unimplemented PRLAR_EL2 register return an UNKNOWN value.
- Writes to unimplemented PRLAR_EL2 register make all PRLAR_EL2 registers value UNKNOWN.

MRS <Xt>, PRLAR_EL2

```

if UInt(PRSELR_EL2.REGION) >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PRLAR_EL2[UInt(AArch64-PRSELR_EL2.REGION)];

```

MSR PRLAR_EL2, <Xt>

```

if UInt(PRSELR_EL2.REGION) >= UInt(MPUIR_EL2.REGION) then
    ConstrainUnpredictableProcedure(Unpredictable_UnimplementedMPURegion_RW_UNKNOWN);
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    PRLAR_EL2[UInt(AArch64-PRSELR_EL2.REGION)] = X[t, 64];

```

A.2.2.99 MAIR_EL2, Memory Attribute Indirection Register (EL2)

Provides the memory attribute encodings corresponding to the possible AttrIdx values in AArch64-PRLAR_EL2 for stage 1 EL2 translation regime and for stage 2 EL1&0 translation regime.

For stage 2 EL1&0 translations, the memory attributes are derived from MAIR_EL2 register as described in the Armv8-R AArch64 architecture.

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

MAIR_EL2 is permitted to be cached in a TLB.

Figure A-159: AARCH64_MAIR_EL2 bit assignments

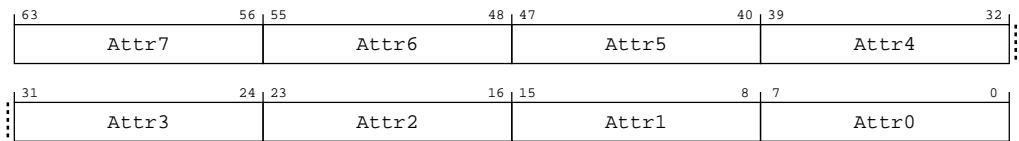


Table A-417: MAIR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	Attr<n>	<p>The memory attribute encoding for an AttrIdx[2:0] gives the value of <n> in Attr<n>.</p> <p>Attr is encoded as follows: Table A-418: Attr description on page 695</p> <p>'dd' is encoded as follows: Table A-419: dd description on page 696</p> <p>'oooo' is encoded as follows: Table A-420: 'oooo' description on page 696</p> <p>R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.</p> <p>'iiii' is encoded as follows: Table A-421: 'iiii' description on page 696</p> <p>R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.</p> <p>The R and W bits in 'oooo' and 'iiii' fields have the following meanings: Table A-422: R or W description on page 696</p>	64 {x}

Table A-418: Attr description

Attr	Meaning
0b0000dd00	Device memory. See encoding of 'dd' for the type of Device memory.
0b0000ddxx, (xx != 00)	UNPREDICTABLE.
0boooooiiii, (oooo != 0000 and iiii != 0000)	Normal memory. See encoding of 'oooo' and 'iiii' for the type of Normal Memory.

Attr	Meaning
0bxxxx0000, where xxxx != 0000	UNPREDICTABLE.

Table A-419: dd description

dd	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Device-nGRE memory
0b11	Device-GRE memory

Table A-420: 'oooo' description

'oooo'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Outer Write-Through Transient
0b0100	Normal memory, Outer Non-cacheable
0b01RW, RW not 0b00	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

Table A-421: 'iiii' description

'iiii'	Meaning
0b0000	See encoding of Attr
0b00RW, RW not 0b00	Normal memory, Inner Write-Through Transient
0b0100	Normal memory, Inner Non-cacheable
0b01RW, RW not 0b00	Normal memory, Inner Write-Back Transient
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

Table A-422: R or W description

R or W	Meaning
0b0	No Allocate
0b1	Allocate

Access

MRS <Xt>, MAIR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b000

MSR MAIR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1010	0b0010	0b000

MRS <Xt>, MAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000

MSR MAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0010	0b000

Accessibility

MRS <Xt>, MAIR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MAIR_EL2;

```

MSR MAIR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    MAIR_EL2 = X[t, 64];

```

MRS <Xt>, MAIR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MAIR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = MAIR_EL1;

```

MSR MAIR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        MAIR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    MAIR_EL1 = X[t, 64];

```

A.2.2.100 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

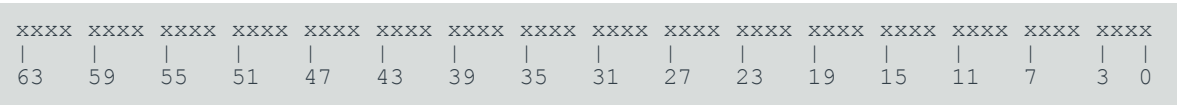
Functional group

Generic System Control

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-160: AARCH64_AMAIR_EL2 bit assignments

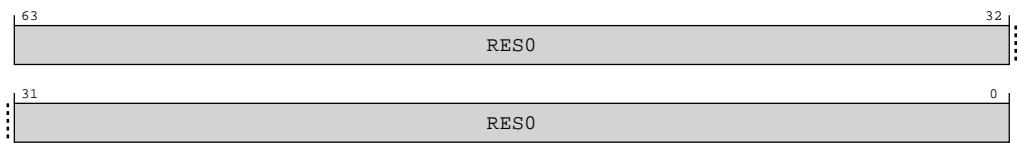


Table A-427: AMAIR_EL2 bit descriptions

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

Access

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

Accessibility

MRS <Xt>, AMAIR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = AMAIR_EL2;
```

MSR AMAIR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AMAIR_EL2 = X[t, 64];
```

A.2.2.101 VBAR_EL2, Vector Base Address Register (EL2)

Holds the vector base address for any exception that is taken to EL2.

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-161: AARCH64_VBAR_EL2 bit assignments

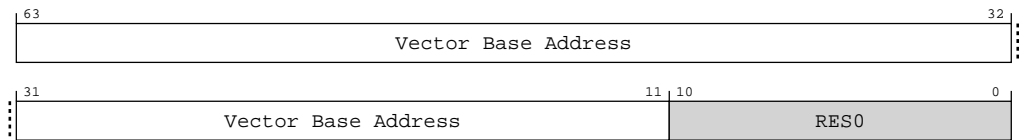


Table A-430: VBAR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:11]	None	Vector Base Address. Base address of the exception vectors for exceptions taken to EL2. Note: <ul style="list-style-type: none">If tagged addresses are being used, bits [55:48] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.If tagged addresses are not being used, bits [63:48] of VBAR_EL2 must be 0 or else the use of the vector address will result in a recursive exception.	53 {x}
[10:0]	RES0	Reserved	RES0

Access

MRS <Xt>, VBAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b000

MSR VBAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b000

Accessibility

MRS <Xt>, VBAR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = VBAR_EL2;
```


MSR VBAR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    VBAR_EL2 = X[t, 64];
```

A.2.2.102 RVBAR_EL2, Reset Vector Base Address Register (if EL3 not implemented)

Contains the **IMPLEMENTATION DEFINED** address that execution starts from after reset.

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	0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-162: AARCH64_RVBAR_EL2 bit assignments

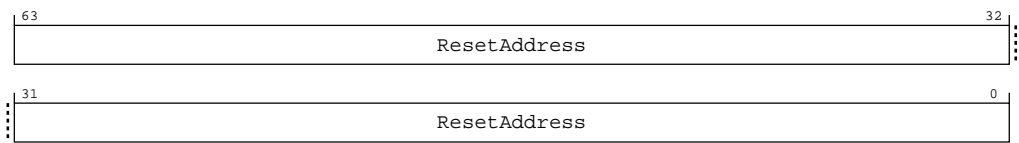


Table A-433: RVBAR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	ResetAddress	The IMPLEMENTATION DEFINED address that execution starts from after reset when executing in 64-bit state. Bits[1:0] of this register are 00, as this address must be aligned, and the address must be within the physical address size supported by the PE. This field takes the value of the CFGRVBARADDRm configuration pins.	64 {x}

Access
MRS <Xt>, RVBAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b001

Accessibility
MRS <Xt>, RVBAR_EL2

```
if PSTATE.EL == EL2 then
    X[t, 64] = RVBAR_EL2;
else
    UNDEFINED;
```

A.2.2.103 RMR_EL2, Reset Management Register (EL2)

A write to the register at EL2 can request a Warm reset.

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	xx0x
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-163: AARCH64_RMR_EL2 bit assignments

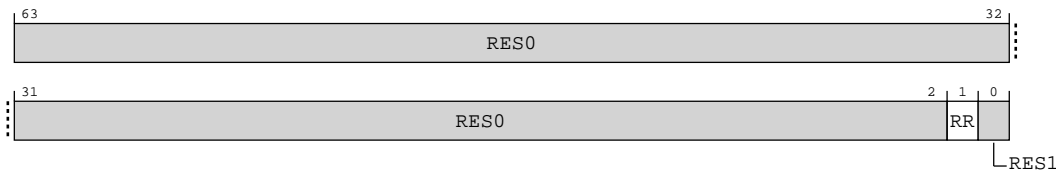


Table A-435: RMR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	RES0
[1]	RR	Reset Request. Setting this bit to 1 requests a Warm reset. 0b0 No effect. 0b1 Warm reset requested.	0b0
[0]	RES1	Reserved	RES1

Access

MRS <Xt>, RMR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b010

MSR RMR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b010

Accessibility

MRS <Xt>, RMR_EL2

```
if PSTATE.EL == EL2 then
    X[t, 64] = RMR_EL2;
else
    UNDEFINED;
```

MSR RMR_EL2, <Xt>

```
if PSTATE.EL == EL2 then
    RMR_EL2 = X[t, 64];
else
    UNDEFINED;
```

A.2.2.104 CONTEXTIDR_EL2, Context ID Register (EL2)

Identifies the current Process Identifier for EL2.

The value of the whole of this register is called the Context ID and is used by:

- The debug logic, for Linked and Unlinked Context ID matching.
- The trace logic, to identify the current process.

The significance of this register is for debug and trace use only.

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-164: AARCH64_CONTEXTIDR_EL2 bit assignments

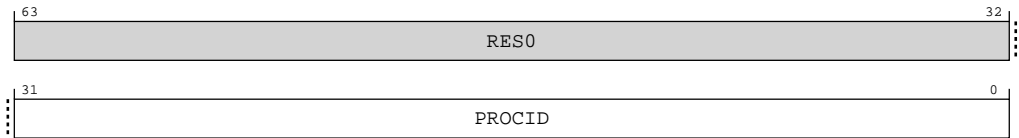


Table A-438: CONTEXTIDR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	PROCID	Process Identifier. This field must be programmed with a unique value that identifies the current process.	32 {x}

Access

MRS <Xt>, CONTEXTIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b001

MSR CONTEXTIDR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b001

Accessibility

MRS <Xt>, CONTEXTIDR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CONTEXTIDR_EL2;
```

MSR CONTEXTIDR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    CONTEXTIDR_EL2 = X[t, 64];
```

A.2.2.105 TPIDR_EL2, EL2 Software Thread ID Register

Provides a location where software executing at EL2 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

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-165: AARCH64_TPIDR_EL2 bit assignments

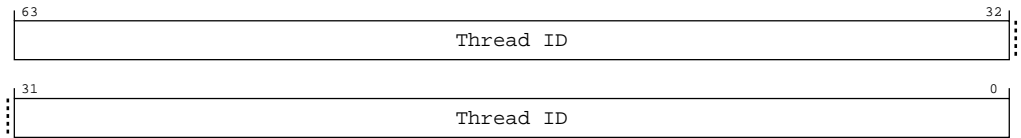


Table A-441: TPIDR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Thread ID. Thread identifying information stored by software running at this Exception level.	64 {x}

Access

MRS <Xt>, TPIDR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b010

MSR TPIDR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1101	0b0000	0b010

Accessibility

MRS <Xt>, TPIDR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TPIDR_EL2;
```

MSR TPIDR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    TPIDR_EL2 = X[t, 64];
```

A.2.2.106 IMP_INTLATENCY_EL2, Interrupt Latency Register

This register provides controls to the Hypervisor to limit certain memory transactions which could negatively impact the processor interrupt latency or the freedom from interference.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic System Control

Access type

See bit descriptions

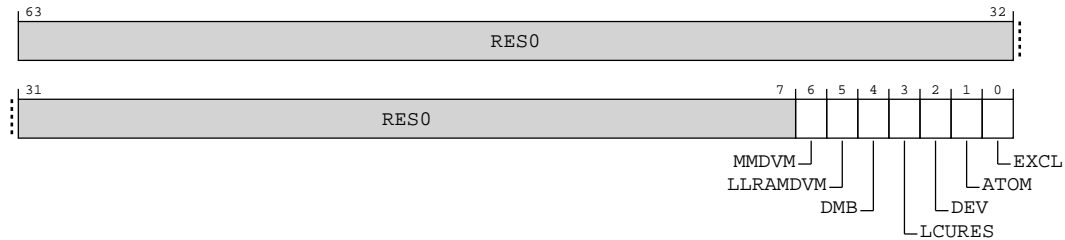
Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x0x0	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-166: AARCH64_IMP_INTLATENCY_EL2 bit assignments**Table A-444: IMP_INTLATENCY_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:7]	RES0	Reserved	RES0
[6]	MMDVM	Limit the behavior of instruction cache maintenance instructions to the MM port. 0b0 No impact on instruction cache maintenance instructions that target the MM port. 0b1 Causes instruction cache maintenance instructions that target the MM port to generate a Data Abort. Note: This includes IC IALLUIS and IC IALLU instructions as these always target the MM port.	0b0
[5]	LLRAMDVM	When CFGLLRAMIMP == 1 Limit the behavior of instruction cache maintenance instructions to the LLRAM port. 0b0 No impact on instruction cache maintenance instructions that target the LLRAM port. 0b1 Causes instruction cache maintenance instructions that target the LLRAM port to generate a Data Abort. Otherwise RES0	'0'
[4]	DMB	Controls the Data Memory Barrier (DMB) instruction behavior. 0b0 This field has no effect on the DMB behavior. 0b1 Forces AArch64-IMP_CPUACTLR_EL1.DMB to behave as if it is 0b1, regardless of the actual value of this bit.	0b0

Bits	Name	Description	Reset
[3]	LCURES	<p>When CFGLLRAMIMP == 1 or CFGSPPIMP == 1</p> <p>Force CPU resources reservation for LCU (LLRAM and SPP) accesses.</p> <p>0b0</p> <p>No impact on CPU resources reservation.</p> <p>0b1</p> <p>Forces AArch64-IMP_CPUACTLR_EL1.LCURES to behave as if it is 0b1, regardless of the actual value of this bit.</p> <p>Otherwise</p> <p>RESO</p>	'0'
[2]	DEV	<p>Limit Device accesses on the MM or LLRAM ports.</p> <p>0b0</p> <p>No impact on Device accesses.</p> <p>0b1</p> <p>Causes any Device access on the MM or LLRAM ports to generate a Data Abort. Also causes any access to the LLPP port which cross a 128-bit boundary or SPP port which cross a 64-bit boundary to generate a Data Abort.</p>	0b0
[1]	ATOM	<p>Limit the behavior of certain atomic accesses:</p> <ul style="list-style-type: none"> All atomic accesses to the MM port. All non-cacheable atomic accesses to the LLRAM port. If CFGLLRAMSHARED == 1, all outer-shareable atomic accesses to the LLRAM port. <p>0b0</p> <p>No impact on atomic accesses.</p> <p>0b1</p> <p>If attributes to the atomic accesses listed above can be changed to execute near, they are forced to do so. Otherwise, causes these far accesses to generate a Data Abort. This setting overrides any behavior selected by AArch64-IMP_CPUACTLR_EL1.ATOM.</p>	0b0
[0]	EXCL	<p>Limit the behavior of certain exclusive accesses:</p> <ul style="list-style-type: none"> All non-cacheable exclusive accesses to the MM port. All non-cacheable exclusive accesses to the LLRAM port. If CFGLLRAMSHARED == 1, all outer-shareable exclusive accesses to the LLRAM port. <p>0b0</p> <p>No impact on exclusive accesses.</p> <p>0b1</p> <p>Causes the exclusive accesses listed above to generate a Data Abort.</p>	0b0

Access

MRS <Xt>, IMP_INTLATENCY_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0001	0b111

MSR IMP_INTLATENCY_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1111	0b0001	0b111

Accessibility

MRS <Xt>, IMP_INTLATENCY_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_INTLATENCY_EL2;

```

MSR IMP_INTLATENCY_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    IMP_INTLATENCY_EL2 = X[t, 64];

```

A.2.3 AArch64 Performance Monitors register description

This section includes the register descriptions for all Performance Monitors registers in the Cortex®-R82 processor.

A.2.3.1 PMINTENSET_EL1, Performance Monitors Interrupt Enable Set Register

Enables the generation of interrupt requests

on overflows from

the cycle counter, AArch64-PMCCNTR_ELO, and the event counters AArch64-PMEVCNTR<n>_ELO. Reading the register shows which overflow interrupt requests

are enabled.

Configurations

AArch64 register PMINTENSET_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.8.11 PMINTENSET_EL1, Performance Monitors Interrupt Enable Set Register](#) on page 2120 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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_PMINTENSET_EL1 bit assignments

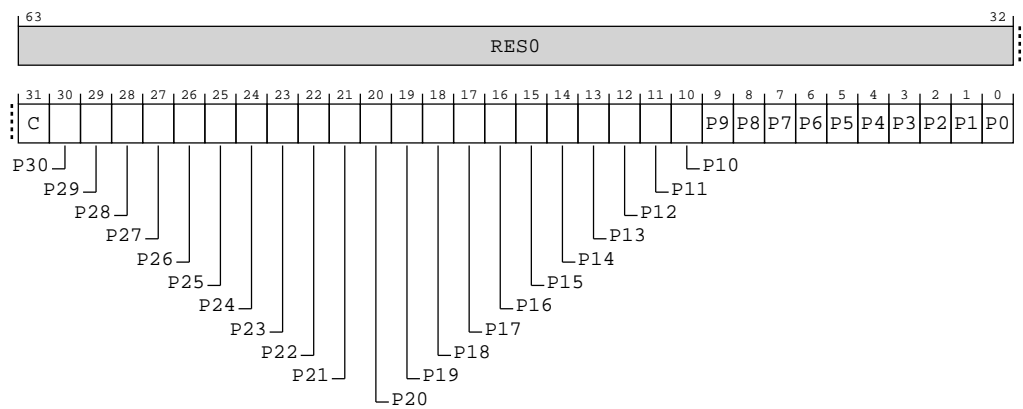


Table A-447: PMINTENSET_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31]	C	<p>Interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO enable. On writes, allows software to enable the interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO. On reads, returns the interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO enable status.</p> <p>0b0</p> <p>Interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO disabled.</p> <p>0b1</p> <p>Interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO enabled.</p> <p>Access to this field is W1S.</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Interrupt request on unsigned overflow of PMEVCNTR<m>_ELO enable. On writes, allows software to enable the interrupt request on unsigned overflow of PMEVCNTR<m>_ELO. On reads, returns the interrupt request on unsigned overflow of PMEVCNTR<m>_ELO enable status.</p> <p>0b0</p> <p>Interrupt request on unsigned overflow of PMEVCNTR<m>_ELO disabled.</p> <p>0b1</p> <p>Interrupt request on unsigned overflow of PMEVCNTR<m>_ELO enabled.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field reads-as-zero and ignores writes if any of the following are true: <ul style="list-style-type: none"> All of the following are true: <ul style="list-style-type: none"> EL2 is implemented and enabled in the current Security state. m >= UInt(AArch64-MDCR_EL2.HPMN). Accessed at EL1. m >= UInt(AArch64-PMCR_ELO.N). Otherwise access to this field is W1S. 	31 {x}

Access

MRS <Xt>, PMINTENSET_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b001

MSR PMINTENSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b001

Accessibility

MRS <Xt>, PMINTENSET_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMINTENSET_EL1;

```

```
elseif PSTATE.EL == EL2 then
    X[t, 64] = PMINTENSET_EL1;
```

MSR PMINTENSET_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMINTENSET_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    PMINTENSET_EL1 = X[t, 64];
```

A.2.3.2 PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear Register

Disables the generation of interrupt requests

on overflows from

the cycle counter, AArch64-PMCCNTR_ELO, and the event counters AArch64-PMEVCNTR<n>_ELO. Reading the register shows which overflow interrupt requests

are enabled.

Configurations

AArch64 register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.8.12 PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear Register](#) on page 2122 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-168: AARCH64_PMINTENCLR_EL1 bit assignments

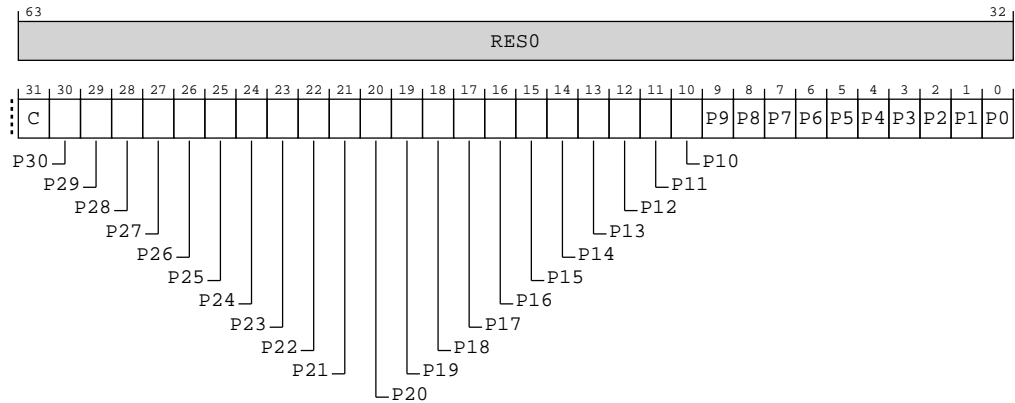


Table A-450: PMINTENCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	<p>Interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO disable. On writes, allows software to disable the interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO. On reads, returns the interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO enable status.</p> <p>0b0</p> <p>Interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO disabled.</p> <p>0b1</p> <p>Interrupt request on unsigned overflow of AArch64-PMCCNTR_ELO enabled.</p> <p>Access to this field is W1C.</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Interrupt request on unsigned overflow of PMEVCNTR<m>_ELO disable. On writes, allows software to disable the interrupt request on unsigned overflow of PMEVCNTR<m>_ELO. On reads, returns the interrupt request on unsigned overflow of PMEVCNTR<m>_ELO enable status.</p> <p>0b0</p> <p>Interrupt request on unsigned overflow of PMEVCNTR<m>_ELO disabled.</p> <p>0b1</p> <p>Interrupt request on unsigned overflow of PMEVCNTR<m>_ELO enabled.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field reads-as-zero and ignores writes if any of the following are true: <ul style="list-style-type: none"> All of the following are true: <ul style="list-style-type: none"> EL2 is implemented and enabled in the current Security state. m >= UInt(AArch64-MDCR_EL2.HPMN). Accessed at EL1. m >= UInt(AArch64-PMCR_ELO.N). Otherwise access to this field is W1C. 	31 {x}

Access

MRS <Xt>, PMINTENCLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b010

MSR PMINTENCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b010

Accessibility

MRS <Xt>, PMINTENCLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMINTENCLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = PMINTENCLR_EL1;
```

MSR PMINTENCLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMINTENCLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    PMINTENCLR_EL1 = X[t, 64];
```

A.2.3.3 PMMIR_EL1, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation to software.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0110	0000	0010	0000	0011
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-169: AARCH64_PMMIR_EL1 bit assignments

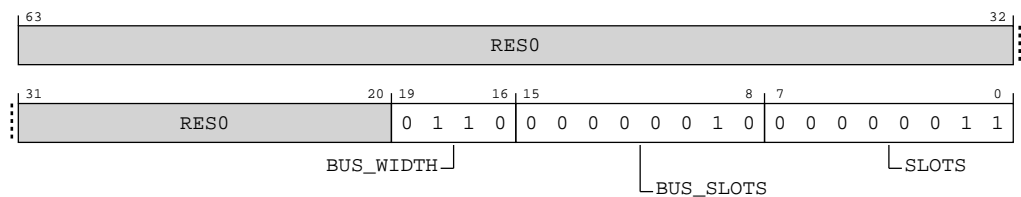


Table A-453: PMMIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0
[19:16]	BUS_WIDTH	Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as $\log_2(\text{number of bytes})$, plus one. 0b0110 32 bytes. All other values are reserved. Each transfer is up to this number of bytes. An access might be smaller than the bus width. When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.	0b0110
[15:8]	BUS_SLOTS	Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle. 0b00000010 The BUS_ACCESS event might increment by at most 2 in a single BUS_CYCLES cycle. When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS. If the bus count information is not available, this field will read as zero.	0x02

Bits	Name	Description	Reset
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle. 0b00000011 STALL_SLOT may increment by up to 3 in a single cycle.	0x03

Access

MRS <Xt>, PMMIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1110	0b110

Accessibility

MRS <Xt>, PMMIR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMMIR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PMMIR_EL1;

```

A.2.3.4 IMP_CLUSTERPMCR_EL1, Cluster Performance Monitors Control Register

Provides details of the cluster Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

AArch64 register IMP_CLUSTERPMCR_EL1 bits [7:0] are architecturally mapped to External register [B.2.2.2.13 CLUSTERPMU_PMCR_EL1, Cluster Performance Monitors Control Register](#) on page 1716 bits [7:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000	0000	0000	0011	0xxx	xxx0	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-170: AARCH64_IMP_CLUSTERPMCR_EL1 bit assignments

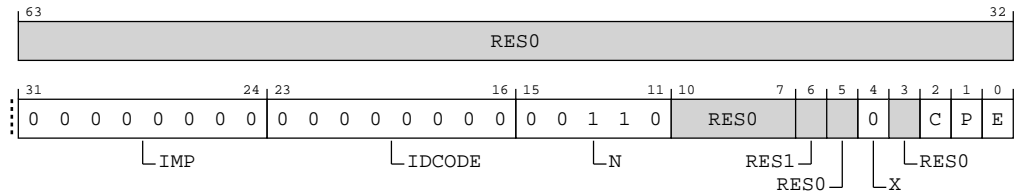


Table A-455: IMP_CLUSTERPMCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	IMP	Implementer code. 0b00000000 No ID information is present. Access to this field is: RO	0x00
[23:16]	IDCODE	Identification code. 0b00000000 No ID information is present. Access to this field is: RO	0x00
[15:11]	N	Indicates the number of event counters implemented. 0b00110 6 counters implemented. Access to this field is: RO	0b00110
[10:7]	RES0	Reserved	RES0
[6]	RES1	Reserved	RES1
[5]	RES0	Reserved	RES0
[4]	X	This field enables the exporting of events over an event bus to another device. 0b0 Cluster PMU events are not exported externally. Access to this field is: RO	0b0
[3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2]	C	<p>Cycle counter reset. This bit is WO. The effects of writing to this bit are:</p> <p>0b0 No action.</p> <p>0b1 Reset AArch64-IMP_CLUSTERPMCCNTR_EL1 to zero.</p> <p>This bit is always RAZ.</p> <p>Note: Resetting AArch64-IMP_CLUSTERPMCCNTR_EL1 does not change the cycle counter overflow bit.</p>	x
[1]	P	<p>Event counter reset. This bit is WO. The effects of writing to this bit are:</p> <p>0b0 No action.</p> <p>0b1 Reset all event counters, not including AArch64-IMP_CLUSTERPMCCNTR_EL1, to zero.</p> <p>This bit is always RAZ.</p> <p>A write of 1 to this bit resets all the event counters.</p> <p>Resetting the event counters does not change the event counter overflow bits.</p>	x
[0]	E	<p>Enable.</p> <p>0b0 All event counters, including AArch64-IMP_CLUSTERPMCCNTR_EL1, are disabled.</p> <p>0b1 All event counters can be enabled by AArch64-IMP_CLUSTERPMCCNTENSET_EL1.</p> <p>This bit is RW.</p>	0b0

Access

MRS <Xt>, IMP_CLUSTERPMCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b000

MSR IMP_CLUSTERPMCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b000

Accessibility

MRS <Xt>, IMP_CLUSTERPMCR_EL1

```

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

```

```
if HCR_EL2.TIDCP == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = IMP_CLUSTERPMCR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMCR_EL1;
```

MSR IMP_CLUSTERPMCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMCR_EL1 = X[t, 64];
```

A.2.3.5 IMP_CLUSTERPMCNTENSET_EL1, Cluster Performance Monitors Count Enable Set register

Enables the Cluster Cycle Count Register, AArch64-IMP_CLUSTERPMCCNTR_EL1, and any implemented event counters. Reading this register shows which counters are enabled.

Configurations

AArch64 register IMP_CLUSTERPMCNTENSET_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.5 CLUSTERPMU_PMCNTENSET_EL1, Cluster Performance Monitors Count Enable Set register](#) on page 1703 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-171: AARCH64_IMP_CLUSTERPMCNTENSET_EL1 bit assignments

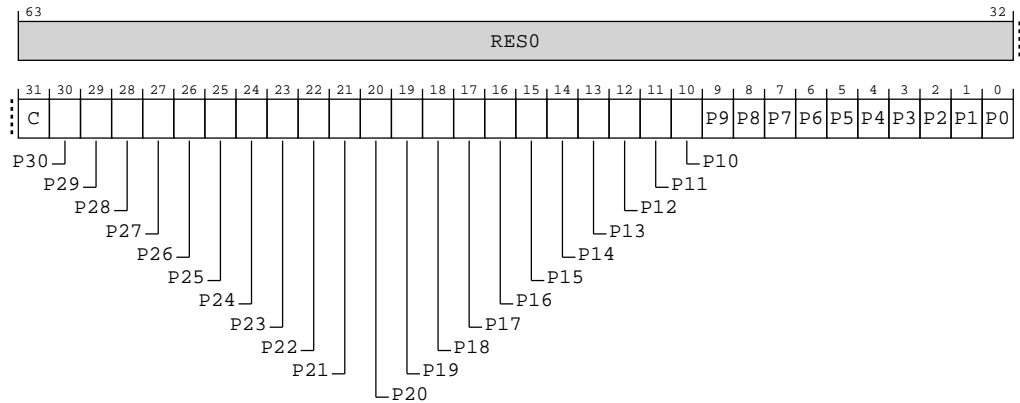


Table A-458: IMP_CLUSTERPMCNTENSET_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	AArch64-IMP_CLUSTERPMCCNTR_EL1 enable bit. Enables the cycle counter register. Possible values are: 0b0 When read, means the cycle counter is disabled. When written, has no effect. 0b1 When read, means the cycle counter is enabled. When written, enables the cycle counter.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter enable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Bits [30:6] are RAZ/WI . 0b0 When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 is disabled. When written, has no effect. 0b1 When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter is enabled. When written, enables ext-CLUSTERPMU_PMEVCNTR<n>_EL1.	31 {x}

Access

MRS <Xt>, IMP_CLUSTERPMCNTENSET_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b001

MSR IMP_CLUSTERPMCNTENSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b001

Accessibility

MRS <Xt>, IMP_CLUSTERPMCNTENSET_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMCNTENSET_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CLUSTERPMCNTENSET_EL1;
```

MSR IMP_CLUSTERPMCNTENSET_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMCNTENSET_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        IMP_CLUSTERPMCNTENSET_EL1 = X[t, 64];
```

A.2.3.6 IMP_CLUSTERPMCNTENCLR_EL1, Cluster Performance Monitors Count Enable Clear register

Disables the Cluster Cycle Count Register, AArch64-IMP_CLUSTERPMCCNTR_EL1, and any implemented event counters. Reading this register shows which counters are enabled.

Configurations

AArch64 register IMP_CLUSTERPMCNTENCLR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.6 CLUSTERPMU_PMCNTENCLR_EL1, Cluster Performance Monitors Count Enable Clear register](#) on page 1705 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

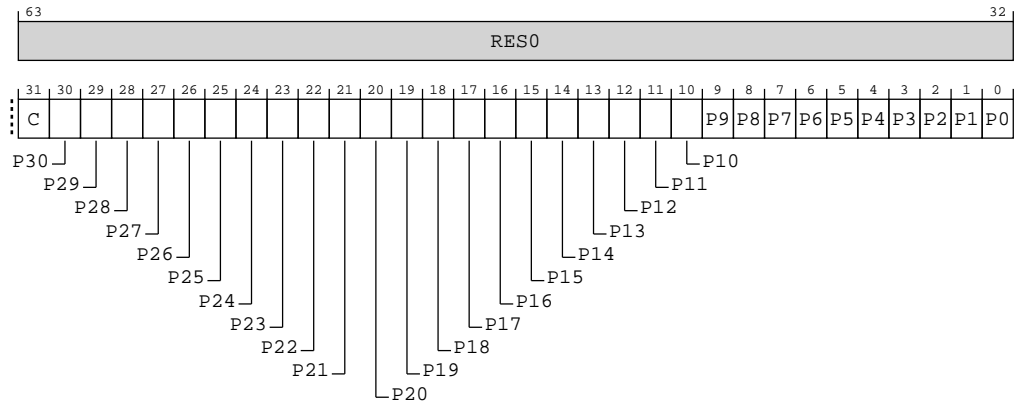
Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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_IMP_CLUSTERPMCNTENCLR_EL1 bit assignments**Table A-461: IMP_CLUSTERPMCNTENCLR_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	AArch64-IMP_CLUSTERPMCNTENCLR_EL1 disable bit. Disables the cycle counter register. Possible values are: 0b0 When read, means the cycle counter is disabled. When written, has no effect. 0b1 When read, means the cycle counter is enabled. When written, disables the cycle counter.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter disable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Bits [30:6] are RAZ/WI . 0b0 When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 is disabled. When written, has no effect. 0b1 When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 is enabled. When written, disables ext-CLUSTERPMU_PMEVCNTR<n>_EL1.	31 {x}

Access

MRS <Xt>, IMP_CLUSTERPMCNTENCLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b010

MSR IMP_CLUSTERPMCNTENCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b010

Accessibility

MRS <Xt>, IMP_CLUSTERPMCNTENCLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMCNTENCLR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMCNTENCLR_EL1;

```

MSR IMP_CLUSTERPMCNTENCLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMCNTENCLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMCNTENCLR_EL1 = X[t, 64];

```

A.2.3.7 IMP_CLUSTERPMOVSSET_EL1, Cluster Performance Monitors Overflow Flag Status Set register

Sets the state of the overflow bit for the Cycle Count Register, AArch64-IMP_CLUSTERPMCCNTR_EL1, and each of the implemented event counters.

Configurations

AArch64 register IMP_CLUSTERPMOVSSET_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.11 CLUSTERPMU_PMOVSSET_EL1, Cluster Performance Monitors Overflow Flag Status Set register](#) on page 1713 bits [31:0].

Attributes**Width**

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-173: AARCH64_IMP_CLUSTERPMOVSSET_EL1 bit assignments

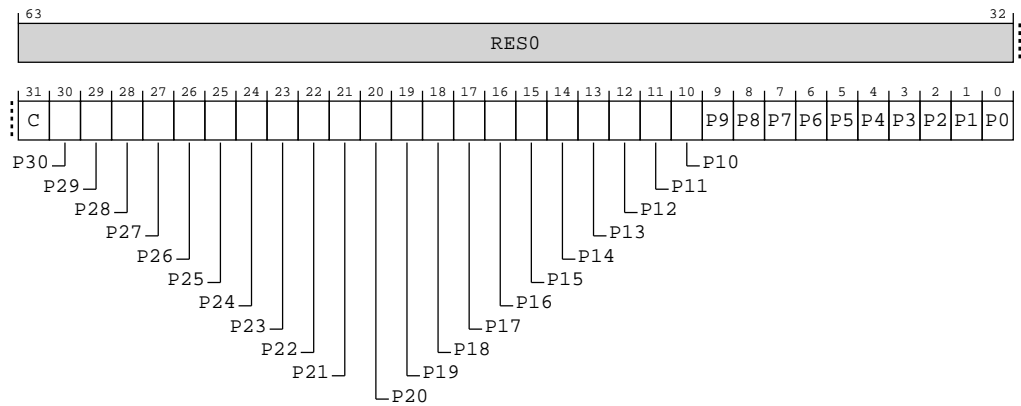


Table A-464: IMP_CLUSTERPMOVSSET_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	<p>Cycle counter overflow set bit.</p> <p>0b0</p> <p>When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter has overflowed since this bit was last cleared. When written, sets the cycle counter overflow bit to 1.</p> <p>Access to this field is W1C.</p>	x

Bits	Name	Description	Reset
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow set bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>Bits [30:6] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has overflowed since this bit was last cleared. When written, sets the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 overflow bit to 1.</p>	31 {x}

Access

MRS <Xt>, IMP_CLUSTERPMOVSSET_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b011

MSR IMP_CLUSTERPMOVSSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b011

Accessibility

MRS <Xt>, IMP_CLUSTERPMOVSSET_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMOVSSET_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMOVSSET_EL1;

```

MSR IMP_CLUSTERPMOVSSET_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMOVSSET_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMOVSSET_EL1 = X[t, 64];

```

A.2.3.8 IMP_CLUSTERPMOVSLR_EL1, Cluster Performance Monitors Overflow Flag Status Clear Register

Contains the state of the overflow bit for the Cluster Cycle Count Register, AArch64-IMP_CLUSTERPMCCNTR_EL1, and each of the implemented event counters. Writing to this register clears these bits.

Configurations

AArch64 register IMP_CLUSTERPMOVSLR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.9 CLUSTERPMU_PMOVSLR_EL1, Cluster Performance Monitors Overflow Flag Status Clear register](#) on page 1710 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-174: AARCH64_IMP_CLUSTERPMOVSLR_EL1 bit assignments

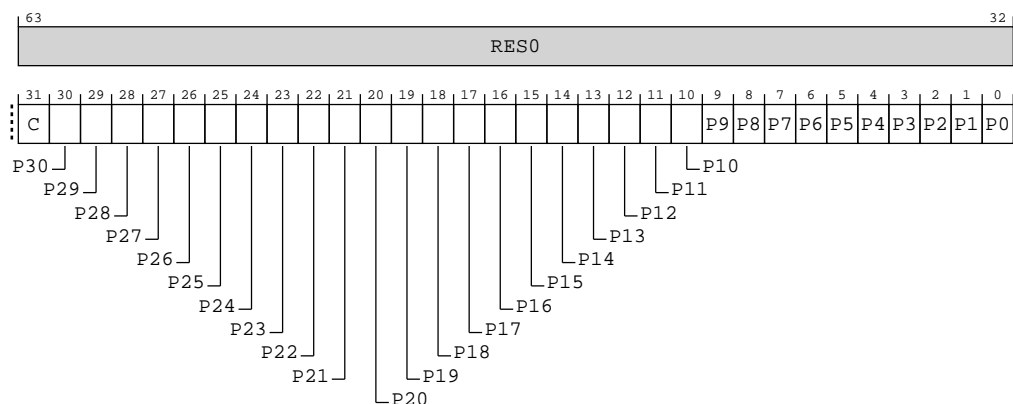


Table A-467: IMP_CLUSTERPMOVSLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	<p>Cycle counter overflow clear bit.</p> <p>0b0</p> <p>When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter has overflowed since this bit was last cleared. When written, clears the cycle counter overflow bit to 0.</p> <p>Access to this field is W1C.</p>	x
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow clear bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>Bits [30:6] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has overflowed since this bit was last cleared. When written, clears the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 overflow bit to 0.</p>	31 {x}

Access

MRS <Xt>, IMP_CLUSTERPMOVSLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b100

MSR IMP_CLUSTERPMOVSLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b100

Accessibility

MRS <Xt>, IMP_CLUSTERPMOVSLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMOVSLR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMOVSLR_EL1;

```

MSR IMP_CLUSTERPMOVSLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMOVSLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMOVSLR_EL1 = X[t, 64];
```

A.2.3.9 IMP_CLUSTERPMSELR_EL1, Cluster Performance Monitors Event Counter Selection Register

Selects the current event counter ext-CLUSTERPMU_PMEVCNTR<n>_EL1 or the cycle counter, CCNT.

IMP_CLUSTERPMSELR_EL1 is used in conjunction with AArch64-IMP_CLUSTERPMXEVTYPER_EL1 to determine the event that increments a selected event counter, and the modes and states in which the selected counter increments.

It is also used in conjunction with AArch64-IMP_CLUSTERPMXEVCNTR_EL1, to determine the value of a selected event counter.

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-175: AARCH64_IMP_CLUSTERPMSELR_EL1 bit assignments

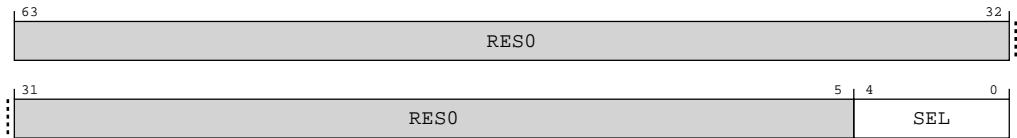


Table A-470: IMP_CLUSTERPMSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	RES0
[4:0]	SEL	<p>Selects event counter, ext-CLUSTERPMU_PMEVCNTR<n>_EL1, where n is the value held in this field. This value identifies which event counter is accessed when a subsequent access to AArch64-IMP_CLUSTERPMXEVTYPER_EL1 or AArch64-IMP_CLUSTERPMXEVCNTR_EL1 occurs.</p> <p>This field can take any value from 0 (0b00000) to (PMCR.N)-1, or 31 (0b11111).</p> <p>When IMP_CLUSTERPMSELR_EL1.SEL is 0b11111, it selects the cycle counter and:</p> <ul style="list-style-type: none">A read of the AArch64-IMP_CLUSTERPMXEVTYPER_EL1 returns the value of AArch64-IMP_CLUSTERPMCCFILTR_EL1.A write of the AArch64-IMP_CLUSTERPMXEVTYPER_EL1 writes to AArch64-IMP_CLUSTERPMCCFILTR_EL1.A read or write of AArch64-IMP_CLUSTERPMXEVCNTR_EL1 has CONSTRAINED UNPREDICTABLE effects. See AArch64-IMP_CLUSTERPMXEVCNTR_EL1 for more details. <p>If this field is set to a value greater than or equal to the number of counters accessible at the current Exception level, but not equal to 31:</p> <ul style="list-style-type: none">Direct reads of this field return an UNKNOWN value.The results of access to AArch64-IMP_CLUSTERPMXEVTYPER_EL1 or AArch64-IMP_CLUSTERPMXEVCNTR_EL1 are CONSTRAINED UNPREDICTABLE. See AArch64-IMP_CLUSTERPMXEVTYPER_EL1 or AArch64-IMP_CLUSTERPMXEVCNTR_EL1 for more details.	5 {x}

Access

MRS <Xt>, IMP_CLUSTERPMSELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b101

MSR IMP_CLUSTERPMSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b101

Accessibility

MRS <Xt>, IMP_CLUSTERPMSELR_EL1

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

```

elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMSELR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMSELR_EL1;

```

MSR IMP_CLUSTERPMSELR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMSELR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMSELR_EL1 = X[t, 64];

```

A.2.3.10 IMP_CLUSTERPMINTENSET_EL1, Cluster Performance Monitors Interrupt Enable Set register

Enables the generation of interrupt requests on overflows from the Cluster Cycle Count Register, AArch64-IMP_CLUSTERPMCCNTR_EL1, and the event counters ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Reading the register shows which overflow interrupt requests are enabled.

Configurations

AArch64 register IMP_CLUSTERPMINTENSET_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.7 CLUSTERPMU_PMINTENSET_EL1, Cluster Performance Monitors Interrupt Enable Set register](#) on page 1706 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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_IMP_CLUSTERPMINTENSET_EL1 bit assignments

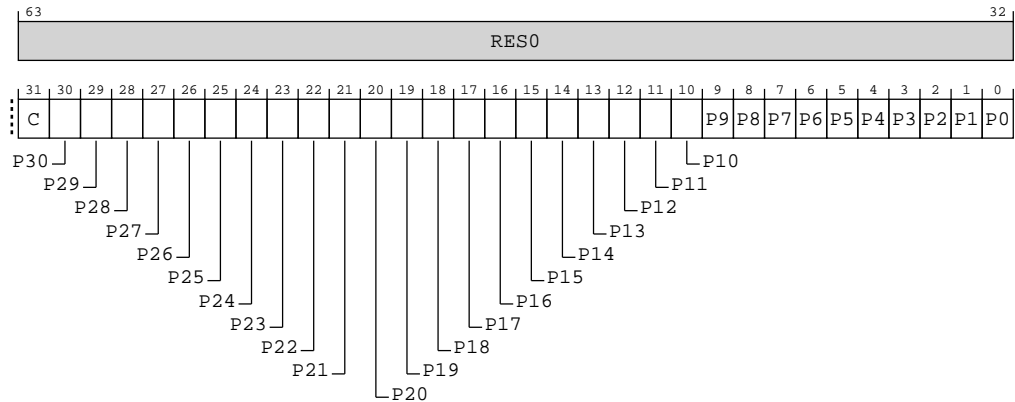


Table A-473: IMP_CLUSTERPMINTENSET_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	<p>AArch64-IMP_CLUSTERPMCCNTR_EL1 overflow interrupt request enable bit. Possible values are:</p> <p>0b0</p> <p>When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.</p>	x
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow interrupt request enable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>Bits [30:6] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is enabled. When written, enables the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 interrupt request.</p>	31 {x}

Access

MRS <Xt>, IMP_CLUSTERPMINTENSET_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b110

MSR IMP_CLUSTERPMINTENSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b110

Accessibility

MRS <Xt>, IMP_CLUSTERPMINTENSET_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMINTENSET_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMINTENSET_EL1;

```

MSR IMP_CLUSTERPMINTENSET_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMINTENSET_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMINTENSET_EL1 = X[t, 64];

```

A.2.3.11 IMP_CLUSTERPMINTENCLR_EL1, Cluster Performance Monitors Interrupt Enable Clear register

Disables the generation of interrupt requests on overflows from the Cluster Cycle Count Register, AArch64-IMP_CLUSTERPMCCNTR_EL1, and the event counters ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Reading the register shows which overflow interrupt requests are enabled.

Configurations

AArch64 register IMP_CLUSTERPMINTENCLR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.8 CLUSTERPMU_PMINTENCLR_EL1, Cluster Performance Monitors Interrupt Enable Clear register](#) on page 1708 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	
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-177: AARCH64_IMP_CLUSTERPMINTENCLR_EL1 bit assignments

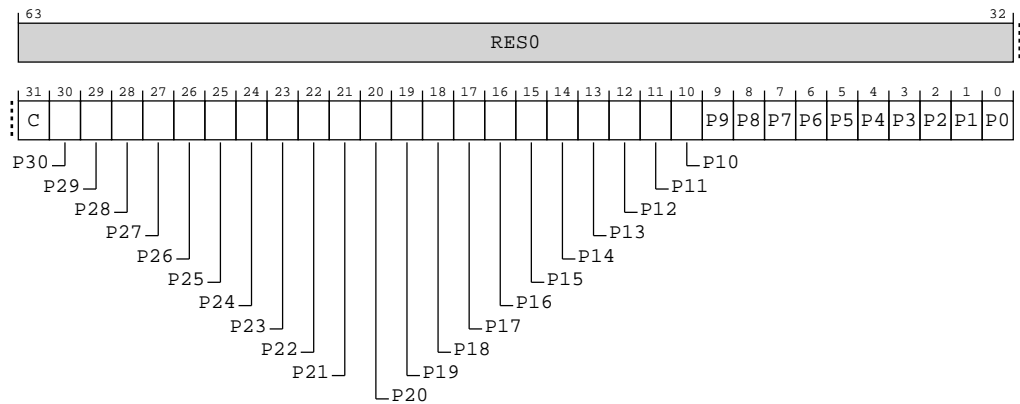


Table A-476: IMP_CLUSTERPMINTENCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	<p>AArch64-IMP_CLUSTERPMCCNTR_EL1 overflow interrupt request disable bit. Possible values are:</p> <p>0b0</p> <p>When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.</p>	x

Bits	Name	Description	Reset
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow interrupt request disable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>Bits [30:6] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is enabled. When written, disables the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 interrupt request.</p>	31 {x}

Access

MRS <Xt>, IMP_CLUSTERPMINTENCLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b111

MSR IMP_CLUSTERPMINTENCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0101	0b111

Accessibility

MRS <Xt>, IMP_CLUSTERPMINTENCLR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMINTENCLR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CLUSTERPMINTENCLR_EL1;

```

MSR IMP_CLUSTERPMINTENCLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMINTENCLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        IMP_CLUSTERPMINTENCLR_EL1 = X[t, 64];

```

A.2.3.12 IMP_CLUSTERPMCCNTR_EL1, Cluster Performance Monitors Cycle Count Register

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See 'Time as measured by the Performance Monitors cycle counter' in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile for more information.

AArch64-IMP_CLUSTERPMCCFILTR_EL1 determines the modes and states in which the IMP_CLUSTERPMCCNTR_EL1 can increment.

Configurations

All counters are subject to any changes in clock frequency, including clock stopping caused by the WFI and WFE instructions. This means that it is CONSTRAINED UNPREDICTABLE whether or not IMP_CLUSTERPMCCNTR_EL1 continues to increment when clocks are stopped by WFI and WFE instructions.

AArch64 register IMP_CLUSTERPMCCNTR_EL1 bits [63:0] are architecturally mapped to External register [B.2.2.2.2 CLUSTERPMU_PMCCNTR_EL1, Cluster Performance Monitors Cycle Counter](#) on page 1698 bits [63:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-178: AARCH64_IMP_CLUSTERPMCCNTR_EL1 bit assignments

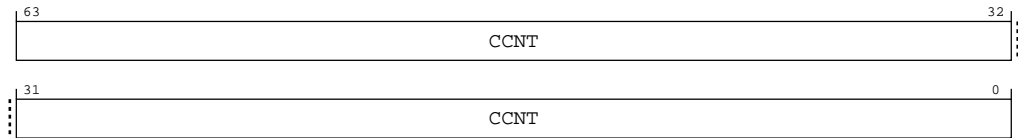


Table A-479: IMP_CLUSTERPMCCNTR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	CCNT	Cycle count. The cycle count increments in every processor clock cycle. Writing 1 to AArch64-IMP_CLUSTERPMCR_EL1.C sets this field to 0.	64 {x}

Access

MRS <Xt>, IMP_CLUSTERPMCCNTR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b000

MSR IMP_CLUSTERPMCCNTR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b000

Accessibility

MRS <Xt>, IMP_CLUSTERPMCCNTR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMCCNTR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMCCNTR_EL1;

```

MSR IMP_CLUSTERPMCCNTR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMCCNTR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMCCNTR_EL1 = X[t, 64];

```

A.2.3.13 IMP_CLUSTERPMXEVTYPER_EL1, Cluster Performance Monitors
Selected Event Type Register

When AArch64-IMP_CLUSTERPMSELR_EL1.SEL selects an event counter, this accesses a ext-CLUSTERPMU_PMEVTYPER<n>_EL1 register. When AArch64-IMP_CLUSTERPMSELR_EL1.SEL selects the cycle counter, this accesses ext-CLUSTERPMU_PMCCFILTR_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

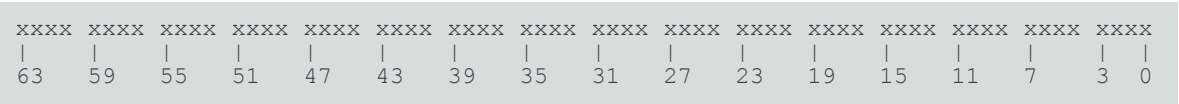
Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-179: AARCH64_IMP_CLUSTERPMXEVTYPER_EL1 bit assignments

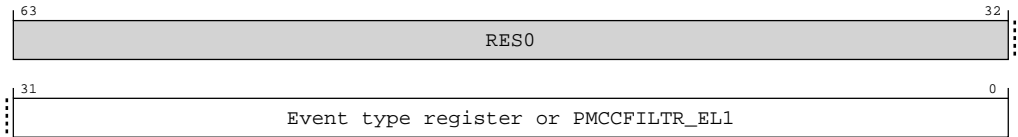


Table A-482: IMP_CLUSTERPMXEVTYPER_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:0]	None	When AArch64-IMP_CLUSTERPMSELR_EL1.SEL == 31, this register accesses ext-CLUSTERPMU_PMCCFILTR_EL1. Otherwise, this register accesses ext-CLUSTERPMU_PMEVTYPER<n>_EL1 where n is the value in AArch64-IMP_CLUSTERPMSELR_EL1.SEL.	32 {x}

Access

If AArch64-IMP_CLUSTERPMSELR_EL1.SEL is greater than or equal to the number of accessible counters then reads and writes of IMP_CLUSTERPMXEVTYPER_EL1 behave as if AArch64-IMP_CLUSTERPMSELR_EL1.SEL has an **UNKNOWN** value less than the number of counters accessible at the current Exception level and Security state.

MRS <Xt>, IMP_CLUSTERPMXEVTYPER_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b001

MSR IMP_CLUSTERPMXEVTYPER_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b001

Accessibility

If AArch64-IMP_CLUSTERPMSELR_EL1.SEL is greater than or equal to the number of accessible counters then reads and writes of IMP_CLUSTERPMXEVTYPER_EL1 behave as if AArch64-IMP_CLUSTERPMSELR_EL1.SEL has an UNKNOWN value less than the number of counters accessible at the current Exception level and Security state.

MRS <Xt>, IMP_CLUSTERPMXEVTYPER_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMXEVTYPER_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMXEVTYPER_EL1;

```

MSR IMP_CLUSTERPMXEVTYPER_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMXEVTYPER_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMXEVTYPER_EL1 = X[t, 64];

```

A.2.3.14 IMP_CLUSTERPMXEVCNTR_EL1, Cluster Performance Monitors Selected Event Count Register

Reads or writes the value of the selected event counter, ext-CLUSTERPMU_PMEVCNTR<n>_EL1. AArch64-IMP_CLUSTERPMSELR_EL1.SEL determines which event counter is selected.

Configurations

This register is available in all configurations.

Attributes

Width

64

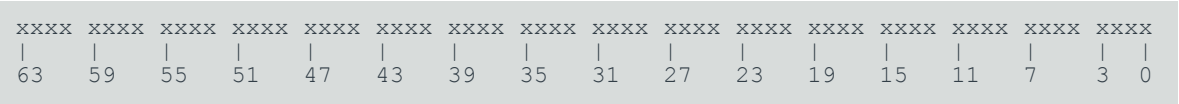
Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-180: AARCH64_IMP_CLUSTERPMXEVCNTR_EL1 bit assignments

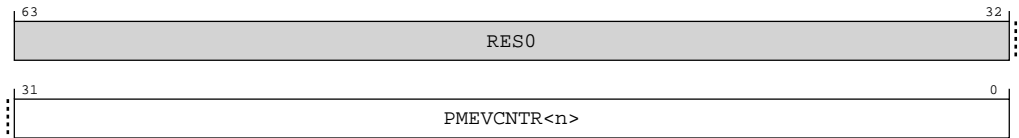


Table A-485: IMP_CLUSTERPMXEVCNTR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	PMEVCNTR<n>	Value of the selected event counter, ext-CLUSTERPMU_PMEVCNTR<n>_EL1, where n is the value stored in AArch64-IMP_CLUSTERPMSELR_EL1.SEL.	32 {x}

Access

If AArch64-IMP_CLUSTERPMSELR_EL1.SEL is greater than or equal to the number of accessible counters then reads and writes of IMP_CLUSTERPMXVCNTR_EL1 behave as if AArch64-IMP_CLUSTERPMSELR_EL1.SEL has an **UNKNOWN** value less than the number of counters accessible at the current Exception level and Security state.

MRS <Xt>, IMP_CLUSTERPMXVCNTR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b010

MSR IMP_CLUSTERPMXVCNTR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b010

Accessibility

If AArch64-IMP_CLUSTERPMSELR_EL1.SEL is greater than or equal to the number of accessible counters then reads and writes of IMP_CLUSTERPMXVCNTR_EL1 behave as if AArch64-IMP_CLUSTERPMSELR_EL1.SEL has an UNKNOWN value less than the number of counters accessible at the current Exception level and Security state.

MRS <Xt>, IMP_CLUSTERPMXVCNTR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMXVCNTR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMXVCNTR_EL1;
```

MSR IMP_CLUSTERPMXVCNTR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMXVCNTR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    IMP_CLUSTERPMXVCNTR_EL1 = X[t, 64];
```

A.2.3.15 IMP_CLUSTERPMCEID0_EL1, Cluster Performance Monitors Common Event Identification register 0

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted.

Configurations

AArch64 register IMP_CLUSTERPMCEID0_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.14 CLUSTERPMU_PMCEID0, Cluster Performance Monitors Common Event Identification register 0](#) on page 1719 bits [31:0].

AArch64 register IMP_CLUSTERPMCEID0_EL1 bits [63:32] are architecturally mapped to External register [B.2.2.2.16 CLUSTERPMU_PMCEID2, Cluster Performance Monitors Common Event Identification register 2](#) on page 1726 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000 0110 0110 0000 0010 0000 0000 0000
0000

Bit descriptions

Figure A-181: AARCH64_IMP_CLUSTERPMCEID0_EL1 bit assignments

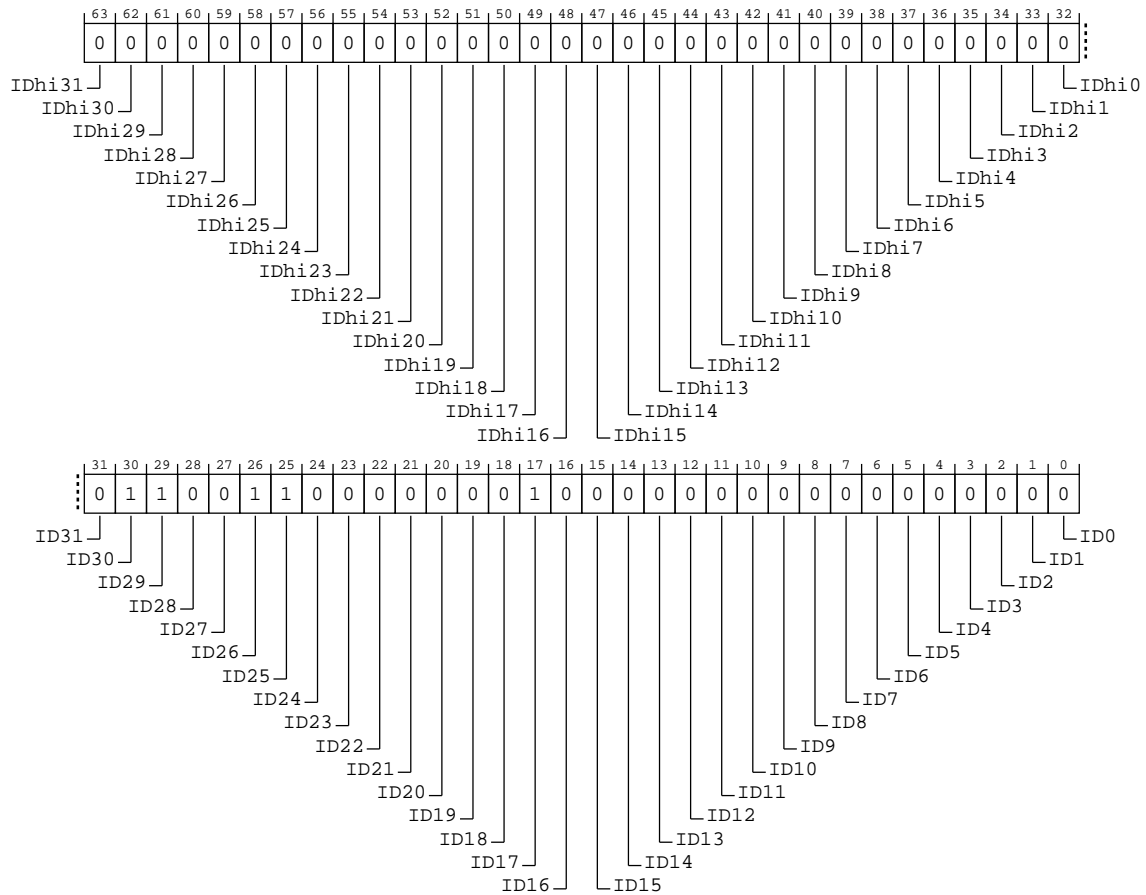


Table A-488: IMP_CLUSTERPMCEID0_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	IDHi31	Common event 0x401F implemented. 0b0 Event 0x401F not implemented.	0b0
[62]	IDHi30	Common event 0x401E implemented. 0b0 Event 0x401E not implemented.	0b0
[61]	IDHi29	Common event 0x401D implemented. 0b0 Event 0x401D not implemented.	0b0
[60]	IDHi28	Common event 0x401C implemented. 0b0 Event 0x401C not implemented.	0b0

Bits	Name	Description	Reset
[59]	IDHi27	Common event 0x401B implemented. 0b0 Event 0x401B not implemented.	0b0
[58]	IDHi26	Common event 0x401A implemented. 0b0 Event 0x401A not implemented.	0b0
[57]	IDHi25	Common event 0x4019 implemented. 0b0 Event 0x4019 not implemented.	0b0
[56]	IDHi24	Common event 0x4018 implemented. 0b0 Event 0x4018 not implemented.	0b0
[55]	IDHi23	Common event 0x4017 implemented. 0b0 Event 0x4017 not implemented.	0b0
[54]	IDHi22	Common event 0x4016 implemented. 0b0 Event 0x4016 not implemented.	0b0
[53]	IDHi21	Common event 0x4015 implemented. 0b0 Event 0x4015 not implemented.	0b0
[52]	IDHi20	Common event 0x4014 implemented. 0b0 Event 0x4014 not implemented.	0b0
[51]	IDHi19	Common event 0x4013 implemented. 0b0 Event 0x4013 not implemented.	0b0
[50]	IDHi18	Common event 0x4012 implemented. 0b0 Event 0x4012 not implemented.	0b0
[49]	IDHi17	Common event 0x4011 implemented. 0b0 Event 0x4011 not implemented.	0b0
[48]	IDHi16	Common event 0x4010 implemented. 0b0 Event 0x4010 not implemented.	0b0
[47]	IDHi15	Common event 0x400F implemented. 0b0 Event 0x400F not implemented.	0b0
[46]	IDHi14	Common event 0x400E implemented. 0b0 Event 0x400E not implemented.	0b0

Bits	Name	Description	Reset
[45]	IDHi13	Common event 0x400D implemented. 0b0 Event 0x400D not implemented.	0b0
[44]	IDHi12	Common event 0x400C implemented. 0b0 Event 0x400C not implemented.	0b0
[43]	IDHi11	Common event 0x400B implemented. 0b0 Event 0x400B not implemented.	0b0
[42]	IDHi10	Common event 0x400A implemented. 0b0 Event 0x400A not implemented.	0b0
[41]	IDHi9	Common event 0x4009 implemented. 0b0 Event 0x4009 not implemented.	0b0
[40]	IDHi8	Common event 0x4008 implemented. 0b0 Event 0x4008 not implemented.	0b0
[39]	IDHi7	Common event 0x4007 implemented. 0b0 Event 0x4007 not implemented.	0b0
[38]	IDHi6	Common event 0x4006 implemented. 0b0 Event 0x4006 not implemented.	0b0
[37]	IDHi5	Common event 0x4005 implemented. 0b0 Event 0x4005 not implemented.	0b0
[36]	IDHi4	Common event 0x4004 implemented. 0b0 Event 0x4004 not implemented.	0b0
[35]	IDHi3	Common event 0x4003 implemented. 0b0 Event 0x4003 not implemented.	0b0
[34]	IDHi2	Common event 0x4002 implemented. 0b0 Event 0x4002 not implemented.	0b0
[33]	IDHi1	Common event 0x4001 implemented. 0b0 Event 0x4001 not implemented.	0b0
[32]	IDHi0	Common event 0x4000 implemented. 0b0 Event 0x4000 not implemented.	0b0

Bits	Name	Description	Reset
[31]	ID31	Common event 0x001F implemented. 0b0 Event 0x001F not implemented.	0b0
[30]	ID30	Common event 0x001E implemented. 0b1 CHAIN event implemented.	0b1
[29]	ID29	Common event 0x001D implemented. 0b1 BUS_CYCLES event implemented.	0b1
[28]	ID28	Common event 0x001C implemented. 0b0 Event 0x001C not implemented.	0b0
[27]	ID27	Common event 0x001B implemented. 0b0 Event 0x001B not implemented.	0b0
[26]	ID26	Common event 0x001A implemented. 0b1 MEMORY_ERROR event implemented.	0b1
[25]	ID25	Common event 0x0019 implemented. 0b1 BUS_ACCESS event implemented.	0b1
[24]	ID24	Common event 0x0018 implemented. 0b0 Event 0x0018 not implemented.	0b0
[23]	ID23	Common event 0x0017 implemented. 0b0 Event 0x0017 not implemented.	0b0
[22]	ID22	Common event 0x0016 implemented. 0b0 Event 0x0016 not implemented.	0b0
[21]	ID21	Common event 0x0015 implemented. 0b0 Event 0x0015 not implemented.	0b0
[20]	ID20	Common event 0x0014 implemented. 0b0 Event 0x0014 not implemented.	0b0
[19]	ID19	Common event 0x0013 implemented. 0b0 Event 0x0013 not implemented.	0b0
[18]	ID18	Common event 0x0012 implemented. 0b0 Event 0x0012 not implemented.	0b0

Bits	Name	Description	Reset
[17]	ID17	Common event 0x0011 implemented. 0b1 CYCLES event implemented.	0b1
[16]	ID16	Common event 0x0010 implemented. 0b0 Event 0x0010 not implemented.	0b0
[15]	ID15	Common event 0x000F implemented. 0b0 Event 0x000F not implemented.	0b0
[14]	ID14	Common event 0x000E implemented. 0b0 Event 0x000E not implemented.	0b0
[13]	ID13	Common event 0x000D implemented. 0b0 Event 0x000D not implemented.	0b0
[12]	ID12	Common event 0x000C implemented. 0b0 Event 0x000C not implemented.	0b0
[11]	ID11	Common event 0x000B implemented. 0b0 Event 0x000B not implemented.	0b0
[10]	ID10	Common event 0x000A implemented. 0b0 Event 0x000A not implemented.	0b0
[9]	ID9	Common event 0x0009 implemented. 0b0 Event 0x0009 not implemented.	0b0
[8]	ID8	Common event 0x0008 implemented. 0b0 Event 0x0008 not implemented.	0b0
[7]	ID7	Common event 0x0007 implemented. 0b0 Event 0x0007 not implemented.	0b0
[6]	ID6	Common event 0x0006 implemented. 0b0 Event 0x0006 not implemented.	0b0
[5]	ID5	Common event 0x0005 implemented. 0b0 Event 0x0005 not implemented.	0b0
[4]	ID4	Common event 0x0004 implemented. 0b0 Event 0x0004 not implemented.	0b0

Bits	Name	Description	Reset
[3]	ID3	Common event 0x0003 implemented. 0b0 Event 0x0003 not implemented.	0b0
[2]	ID2	Common event 0x0002 implemented. 0b0 Event 0x0002 not implemented.	0b0
[1]	ID1	Common event 0x0001 implemented. 0b0 Event 0x0001 not implemented.	0b0
[0]	ID0	Common event 0x0000 implemented. 0b0 Event 0x0000 not implemented.	0b0

Access

MRS <Xt>, IMP_CLUSTERPMCEID0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b100

Accessibility

MRS <Xt>, IMP_CLUSTERPMCEID0_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMCEID0_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = IMP_CLUSTERPMCEID0_EL1;

```

A.2.3.16 IMP_CLUSTERPMCEID1_EL1, Cluster 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 0x0000 to 0x001F and 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted.

Configurations

AArch64 register IMP_CLUSTERPMCEID1_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.15 CLUSTERPMU_PMCEID1, Cluster Performance Monitors Common Event Identification register 1](#) on page 1722 bits [31:0].

AArch64 register IMP_CLUSTERPMCEID1_EL1 bits [63:32] are architecturally mapped to External register [B.2.2.2.17 CLUSTERPMU_PMCEID3, Cluster Performance Monitors Common Event Identification register 3](#) on page 1730 bits [31:0].

Attributes

Width

64

Functional group

Performance 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
0000

Bit descriptions

Figure A-182: AARCH64_IMP_CLUSTERPMCEID1_EL1 bit assignments

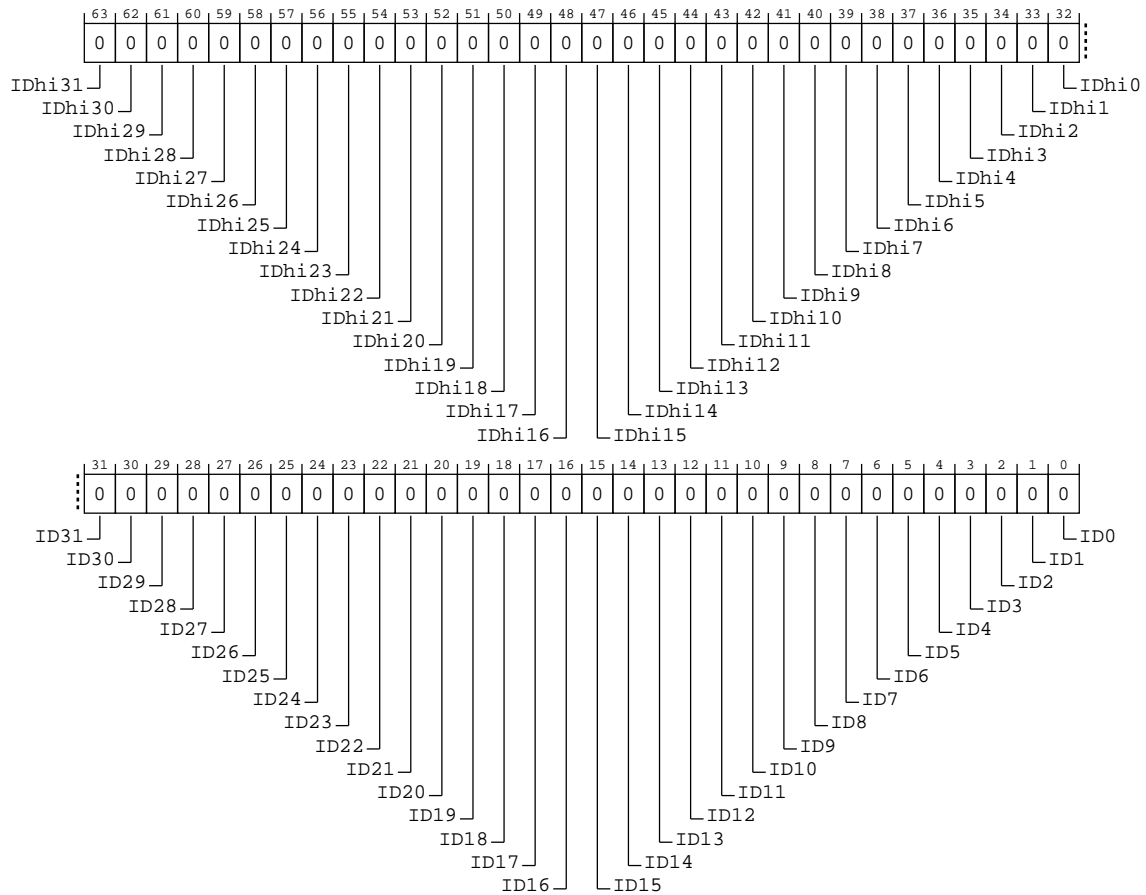


Table A-490: IMP_CLUSTERPMCEID1_EL1 bit descriptions

Bits	Name	Description	Reset
[63]	IDHi31	Common event 0x403F implemented. 0b0 Event 0x403F not implemented.	0b0
[62]	IDHi30	Common event 0x403E implemented. 0b0 Event 0x403E not implemented.	0b0
[61]	IDHi29	Common event 0x403D implemented. 0b0 Event 0x403D not implemented.	0b0
[60]	IDHi28	Common event 0x403C implemented. 0b0 Event 0x403C not implemented.	0b0

Bits	Name	Description	Reset
[59]	IDHi27	Common event 0x403B implemented. 0b0 Event 0x403B not implemented.	0b0
[58]	IDHi26	Common event 0x403A implemented. 0b0 Event 0x403A not implemented.	0b0
[57]	IDHi25	Common event 0x4039 implemented. 0b0 Event 0x4039 not implemented.	0b0
[56]	IDHi24	Common event 0x4038 implemented. 0b0 Event 0x4038 not implemented.	0b0
[55]	IDHi23	Common event 0x4037 implemented. 0b0 Event 0x4037 not implemented.	0b0
[54]	IDHi22	Common event 0x4036 implemented. 0b0 Event 0x4036 not implemented.	0b0
[53]	IDHi21	Common event 0x4035 implemented. 0b0 Event 0x4035 not implemented.	0b0
[52]	IDHi20	Common event 0x4034 implemented. 0b0 Event 0x4034 not implemented.	0b0
[51]	IDHi19	Common event 0x4033 implemented. 0b0 Event 0x4033 not implemented.	0b0
[50]	IDHi18	Common event 0x4032 implemented. 0b0 Event 0x4032 not implemented.	0b0
[49]	IDHi17	Common event 0x4031 implemented. 0b0 Event 0x4031 not implemented.	0b0
[48]	IDHi16	Common event 0x4030 implemented. 0b0 Event 0x4030 not implemented.	0b0
[47]	IDHi15	Common event 0x402F implemented. 0b0 Event 0x402F not implemented.	0b0
[46]	IDHi14	Common event 0x402E implemented. 0b0 Event 0x402E not implemented.	0b0

Bits	Name	Description	Reset
[45]	IDHi13	Common event 0x402D implemented. 0b0 Event 0x402D not implemented.	0b0
[44]	IDHi12	Common event 0x402C implemented. 0b0 Event 0x402C not implemented.	0b0
[43]	IDHi11	Common event 0x402B implemented. 0b0 Event 0x402B not implemented.	0b0
[42]	IDHi10	Common event 0x402A implemented. 0b0 Event 0x402A not implemented.	0b0
[41]	IDHi9	Common event 0x4029 implemented. 0b0 Event 0x4029 not implemented.	0b0
[40]	IDHi8	Common event 0x4028 implemented. 0b0 Event 0x4028 not implemented.	0b0
[39]	IDHi7	Common event 0x4027 implemented. 0b0 Event 0x4027 not implemented.	0b0
[38]	IDHi6	Common event 0x4026 implemented. 0b0 Event 0x4026 not implemented.	0b0
[37]	IDHi5	Common event 0x4025 implemented. 0b0 Event 0x4025 not implemented.	0b0
[36]	IDHi4	Common event 0x4024 implemented. 0b0 Event 0x4024 not implemented.	0b0
[35]	IDHi3	Common event 0x4023 implemented. 0b0 Event 0x4023 not implemented.	0b0
[34]	IDHi2	Common event 0x4022 implemented. 0b0 Event 0x4022 not implemented.	0b0
[33]	IDHi1	Common event 0x4021 implemented. 0b0 Event 0x4021 not implemented.	0b0
[32]	IDHi0	Common event 0x4020 implemented. 0b0 Event 0x4020 not implemented.	0b0

Bits	Name	Description	Reset
[31]	ID31	Common event 0x003F implemented. 0b0 Event 0x003F not implemented.	0b0
[30]	ID30	Common event 0x003E implemented. 0b0 Event 0x003E not implemented.	0b0
[29]	ID29	Common event 0x003D implemented. 0b0 Event 0x003D not implemented.	0b0
[28]	ID28	Common event 0x003C implemented. 0b0 Event 0x003C not implemented.	0b0
[27]	ID27	Common event 0x003B implemented. 0b0 Event 0x003B not implemented.	0b0
[26]	ID26	Common event 0x003A implemented. 0b0 Event 0x003A not implemented.	0b0
[25]	ID25	Common event 0x0039 implemented. 0b0 Event 0x0039 not implemented.	0b0
[24]	ID24	Common event 0x0038 implemented. 0b0 Event 0x0038 not implemented.	0b0
[23]	ID23	Common event 0x0037 implemented. 0b0 Event 0x0037 not implemented.	0b0
[22]	ID22	Common event 0x0036 implemented. 0b0 Event 0x0036 not implemented.	0b0
[21]	ID21	Common event 0x0035 implemented. 0b0 Event 0x0035 not implemented.	0b0
[20]	ID20	Common event 0x0034 implemented. 0b0 Event 0x0034 not implemented.	0b0
[19]	ID19	Common event 0x0033 implemented. 0b0 Event 0x0033 not implemented.	0b0
[18]	ID18	Common event 0x0032 implemented. 0b0 Event 0x0032 not implemented.	0b0

Bits	Name	Description	Reset
[17]	ID17	Common event 0x0031 implemented. 0b0 Event 0x0031 not implemented.	0b0
[16]	ID16	Common event 0x0030 implemented. 0b0 Event 0x0030 not implemented.	0b0
[15]	ID15	Common event 0x002F implemented. 0b0 Event 0x002F not implemented.	0b0
[14]	ID14	Common event 0x002E implemented. 0b0 Event 0x002E not implemented.	0b0
[13]	ID13	Common event 0x002D implemented. 0b0 Event 0x002D not implemented.	0b0
[12]	ID12	Common event 0x002C implemented. 0b0 Event 0x002C not implemented.	0b0
[11]	ID11	Common event 0x002B implemented. 0b0 Event 0x002B not implemented.	0b0
[10]	ID10	Common event 0x002A implemented. 0b0 Event 0x002A not implemented.	0b0
[9]	ID9	Common event 0x0029 implemented. 0b0 Event 0x0029 not implemented.	0b0
[8]	ID8	Common event 0x0028 implemented. 0b0 Event 0x0028 not implemented.	0b0
[7]	ID7	Common event 0x0027 implemented. 0b0 Event 0x0027 not implemented.	0b0
[6]	ID6	Common event 0x0026 implemented. 0b0 Event 0x0026 not implemented.	0b0
[5]	ID5	Common event 0x0025 implemented. 0b0 Event 0x0025 not implemented.	0b0
[4]	ID4	Common event 0x0024 implemented. 0b0 Event 0x0024 not implemented.	0b0

Bits	Name	Description	Reset
[3]	ID3	Common event 0x0023 implemented. 0b0 Event 0x0023 not implemented.	0b0
[2]	ID2	Common event 0x0022 implemented. 0b0 Event 0x0022 not implemented.	0b0
[1]	ID1	Common event 0x0021 implemented. 0b0 Event 0x0021 not implemented.	0b0
[0]	ID0	Common event 0x0020 implemented. 0b0 Event 0x0020 not implemented.	0b0

Access

MRS <Xt>, IMP_CLUSTERPMCEID1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b101

Accessibility

MRS <Xt>, IMP_CLUSTERPMCEID1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMCEID1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMCEID1_EL1;

```

A.2.3.17 IMP_CLUSTERPMCLAIMSET_EL1, Cluster Performance Monitors Claim Set register

Used by software to set CLAIM bits to 1.

Configurations

AArch64 register IMP_CLUSTERPMCLAIMSET_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.20 CLUSTERPMU_PMCLAIMSET_EL1, Cluster Performance Monitors Claim Set register](#) on page 1736 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	
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-183: AARCH64_IMP_CLUSTERPMCLAIMSET_EL1 bit assignments

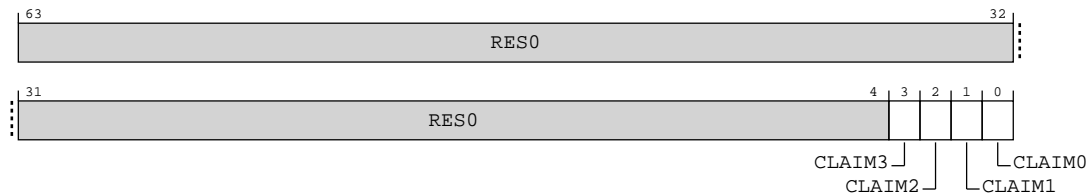


Table A-492: IMP_CLUSTERPMCLAIMSET_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:0]	CLAIM<x>, bit[x], where x = 3 to 0	CLAIM tag set bit. 0b0 No action. 0b1 Indirectly set claim bit to 1.	xxxxx ¹⁹

Access

MRS <Xt>, IMP_CLUSTERPMCLAIMSET_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b110

MSR IMP_CLUSTERPMCLAIMSET_EL1, <Xt>

¹⁹ An External Debug reset clears the CLAIM tag bits to 0.

Copyright © 2022–2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b110

Accessibility

MRS <Xt>, IMP_CLUSTERPMCLAIMSET_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = IMP_CLUSTERPMCLAIMSET_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMCLAIMSET_EL1;

```

MSR IMP_CLUSTERPMCLAIMSET_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMCLAIMSET_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    IMP_CLUSTERPMCLAIMSET_EL1 = X[t, 64];

```

A.2.3.18 IMP_CLUSTERPMCLAIMCLR_EL1, Cluster Performance Monitors Claim Clear register

Used by software to read the values of the CLAIM bits, and to clear these bits to 0.

Configurations

AArch64 register IMP_CLUSTERPMCLAIMCLR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.2.21 CLUSTERPMU_PMCLAIMCLR_EL1, Cluster Performance Monitors Claim Clear register](#) on page 1737 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions


Reset value

```

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

```

6359555147433935312723191511730



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-184: AARCH64_IMP_CLUSTERPMCLAIMCLR_EL1 bit assignments

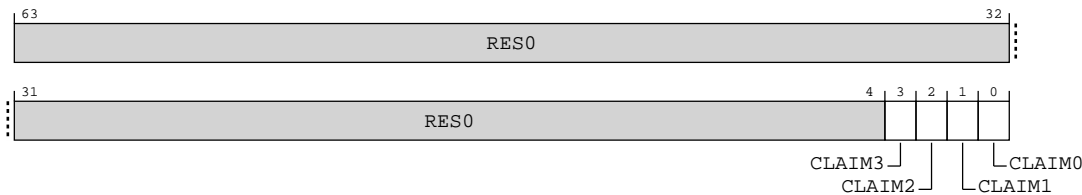


Table A-495: IMP_CLUSTERPMCLAIMCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:0]	CLAIM<x>, bit[x], where x = 3 to 0	CLAIM tag clear bit. 0b0 No action. 0b1 Indirectly clear claim bit to 0.	xxxx ²⁰

Access

MRS <Xt>, IMP_CLUSTERPMCLAIMCLR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b111

MSR IMP_CLUSTERPMCLAIMCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1111	0b0110	0b111

Accessibility

MRS <Xt>, IMP_CLUSTERPMCLAIMCLR_EL1

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

²⁰ An External Debug reset clears the CLAIM tag bits to 0.

Copyright © 2022–2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
if HCR_EL2.TIDCP == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = IMP_CLUSTERPMCLAIMCLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = IMP_CLUSTERPMCLAIMCLR_EL1;
```

MSR IMP_CLUSTERPMCLAIMCLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CLPMU == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_CLUSTERPMCLAIMCLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    IMP_CLUSTERPMCLAIMCLR_EL1 = X[t, 64];
```

A.2.3.19 PMCR_ELO, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

AArch64 register PMCR_ELO bits [63:32, 10:0] are architecturally mapped to External register [B.2.2.8.17 PMCR_ELO, Performance Monitors Control Register](#) on page 2132 bits [63:32].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

RAZW

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000	0000	0000	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-185: AARCH64_PMCR_ELO bit assignments

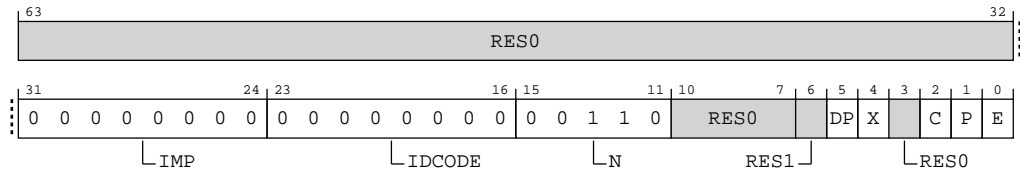


Table A-498: PMCR_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	IMP	<p>Implementer code. PMCR_ELO.IDCODE is RES0 and software must use AArch64-MIDR_EL1 to identify the PE.</p> <p>0b00000000</p> <p>No ID information is present.</p> <p>Use of this field is deprecated.</p>	0x00
[23:16]	IDCODE	<p>Identification code.</p> <p>0b00000000</p> <p>No ID information is present.</p>	0x00
[15:11]	N	<p>Indicates the number of event counters implemented. Reads of this field from EL1 and EL0 return the value of AArch64-MDCR_EL2.HPMN.</p> <p>0b00110</p> <p>6 counters implemented.</p>	0b00110
[10:7]	RES0	Reserved	RES0
[6]	RES1	Reserved	RES1
[5]	DP	<p>Disable cycle counter when event counting is prohibited.</p> <p>0b0</p> <p>Cycle counting by AArch64-PMCCNTR_ELO is not affected by this mechanism.</p> <p>0b1</p> <p>Cycle counting by AArch64-PMCCNTR_ELO is disabled in prohibited regions and when event counting is frozen:</p> <ul style="list-style-type: none"> If FEAT_PMUv3p1 is implemented, EL2 is implemented, and AArch64-MDCR_EL2.HPMD is 1, then cycle counting by AArch64-PMCCNTR_ELO is disabled at EL2. <p>The conditions when this field disables the cycle counter are the same as when event counting by an event counter AArch64-PMEVCNTR<n>_ELO is prohibited or frozen, when either EL2 is not implemented or n is less than AArch64-MDCR_EL2.HPMN.</p> <p>For more information, see <i>Prohibiting event and cycle counting</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x

Bits	Name	Description	Reset
[4]	X	<p>Enable export of events to the core trace unit (ETM).</p> <p>0b0 Do not export events.</p> <p>0b1 Export events to the ETM, where not prohibited.</p>	0b0
[3]	RES0	Reserved	RES0
[2]	C	<p>Cycle counter reset. The effects of writing to this bit are:</p> <p>0b0 No action.</p> <p>0b1 Reset AArch64-PMCCNTR_ELO to zero.</p> <p>Resetting AArch64-PMCCNTR_ELO does not change the cycle counter overflow bit.</p> <p>Access to this field is: WO/RAZ</p>	0b0
[1]	P	<p>Event counter reset.</p> <p>In the description of this field, PMN is PMCR_ELO.N.</p> <p>0b0 No action.</p> <p>0b1 If n is in the range of affected event counters, resets each event counter AArch64-PMEVCNTR<n>_ELO to zero.</p> <p>The effects of writing to this bit are:</p> <ul style="list-style-type: none"> • If EL2 is implemented and enabled in the current Security state, in EL0 and EL1, if PMN is not 0, a write of 1 to this bit resets event counters in the range [0 .. (PMN-1)]. • In EL2, a write of 1 to this bit resets all the event counters. • This field does not affect the operation of other event counters and AArch64-PMCCNTR_ELO. <p>Resetting the event counters does not change the event counter overflow bits.</p> <p>Access to this field is: WO/RAZ</p>	0b0

Bits	Name	Description	Reset
[0]	E	<p>Enable.</p> <p>In the description of this field:</p> <ul style="list-style-type: none"> If EL2 is implemented, then PMN is AArch64-MDCR_EL2.HPMN. If EL2 is not implemented, then PMN is PMCR_ELO.N. <p>0b0</p> <p>Affected counters are disabled and do not count.</p> <p>0b1</p> <p>Affected counters are enabled by AArch64-PMCNTENSET_ELO.</p> <p>The counters affected by this field are:</p> <ul style="list-style-type: none"> If EL2 is implemented, event counters AArch64-PMEVCNTR<n>_ELO for values of n less than PMN. This applies even when EL2 is disabled in the current Security state. If EL2 is not implemented, all event counters AArch64-PMEVCNTR<n>_ELO. The cycle counter AArch64-PMCCNTR_ELO. <p>Other event counters are not affected by this field.</p>	0b0

Access

MRS <Xt>, PMCR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

MSR PMCR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

Accessibility

MRS <Xt>, PMCR_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_ELO.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCR_ELO;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCR_ELO;

```

```
elseif PSTATE.EL == EL2 then
    X[t, 64] = PMCR_EL0;
```

MSR PMCR_EL0, <Xt>

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    PMCR_EL0 = X[t, 64];
```

A.2.3.20 PMCNTENSET_EL0, Performance Monitors Count Enable Set Register

Enables the Cycle Count Register, AArch64-PMCCNTR_EL0, and any implemented event counters AArch64-PMEVCNTR<n>_EL0. Reading this register shows which counters are enabled.

Configurations

AArch64 register PMCNTENSET_EL0 bits [31:0] are architecturally mapped to External register [B.2.2.8.9 PMCNTENSET_EL0, Performance Monitors Count Enable Set Register](#) on page 2117 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-186: AARCH64_PMCNTENSET_ELO bit assignments

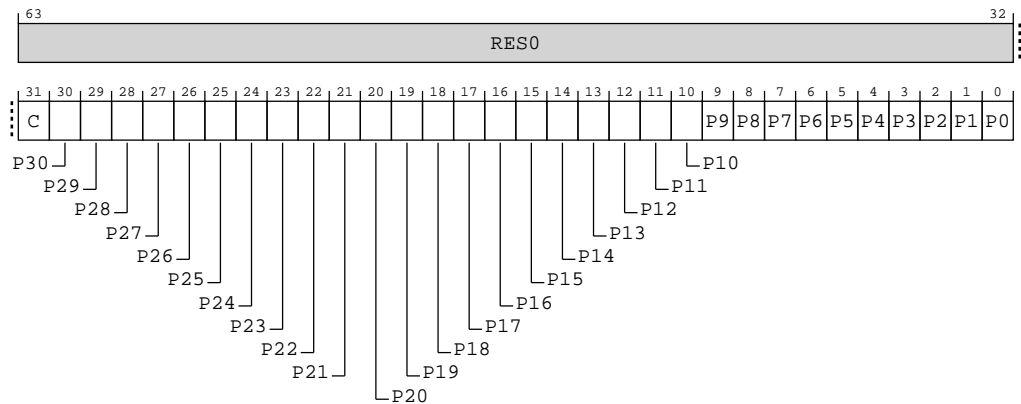


Table A-501: PMCNTENSET_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	AArch64-PMCCNTR_ELO enable. On writes, allows software to enable AArch64-PMCCNTR_ELO. On reads, returns the AArch64-PMCCNTR_ELO enable status. 0b0 AArch64-PMCCNTR_ELO disabled. 0b1 AArch64-PMCCNTR_ELO enabled. Access to this field is W1S.	x

Bits	Name	Description	Reset
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>PMEVCNTR<m>_ELO enable. On writes, allows software to enable PMEVCNTR<m>_ELO. On reads, returns the PMEVCNTR<m>_ELO enable status.</p> <p>0b0</p> <p>PMEVCNTR<m>_ELO disabled.</p> <p>0b1</p> <p>PMEVCNTR<m>_ELO enabled.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field reads-as-zero and ignores writes if any of the following are true: <ul style="list-style-type: none"> All of the following are true: <ul style="list-style-type: none"> EL2 is implemented and enabled in the current Security state. m >= UInt(AArch64-MDCR_EL2.HPMN). Accessed at EL0 or EL1. m >= UInt(AArch64-PMCR_ELO.N). Otherwise access to this field is W1S. 	31 {x}

Access

MRS <Xt>, PMCNTENSET_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b001

MSR PMCNTENSET_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b001

Accessibility

MRS <Xt>, PMCNTENSET_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCNTENSET_ELO;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCNTENSET_ELO;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = PMCNTENSET_ELO;

```

MSR PMCNTENSET_ELO, <Xt>

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMCNTENSET_ELO = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMCNTENSET_ELO = X[t, 64];
    elsif PSTATE.EL == EL2 then
        PMCNTENSET_ELO = X[t, 64];
```

A.2.3.21 PMCNTENCLR_ELO, Performance Monitors Count Enable Clear Register

Disables the Cycle Count Register, AArch64-PMCCNTR_ELO, and any implemented event counters AArch64-PMEVCNTR<n>_ELO. Reading this register shows which counters are enabled.

Configurations

AArch64 register PMCNTENCLR_ELO bits [31:0] are architecturally mapped to External register [B.2.2.8.10 PMCNTENCLR_ELO, Performance Monitors Count Enable Clear Register](#) on page 2119 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-187: AARCH64_PMCNTENCLR_ELO bit assignments

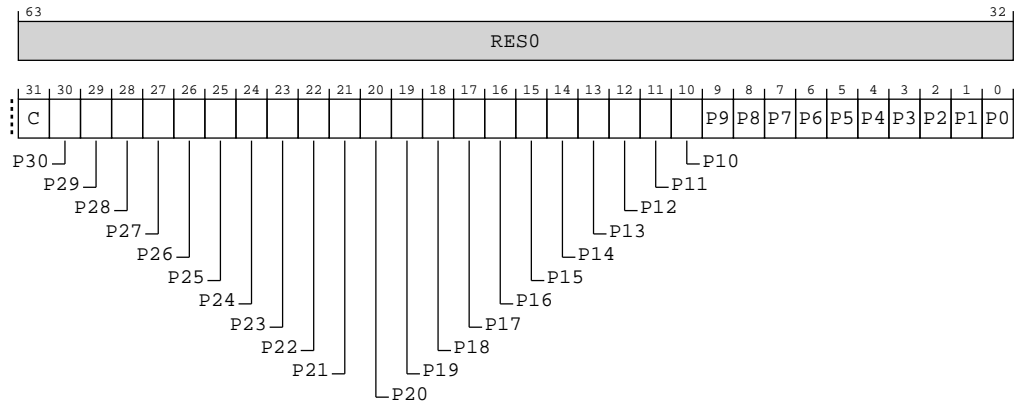


Table A-504: PMCNTENCLR_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	<p>AArch64-PMCCNTR_ELO disable. On writes, allows software to disable AArch64-PMCCNTR_ELO. On reads, returns the AArch64-PMCCNTR_ELO enable status.</p> <p>0b0</p> <p>AArch64-PMCCNTR_ELO disabled.</p> <p>0b1</p> <p>AArch64-PMCCNTR_ELO enabled.</p> <p>Access to this field is W1C.</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>PMEVCNTR<m>_ELO disable. On writes, allows software to disable PMEVCNTR<m>_ELO. On reads, returns the PMEVCNTR<m>_ELO enable status.</p> <p>0b0</p> <p>PMEVCNTR<m>_ELO disabled.</p> <p>0b1</p> <p>PMEVCNTR<m>_ELO enabled.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field reads-as-zero and ignores writes if any of the following are true: <ul style="list-style-type: none"> All of the following are true: <ul style="list-style-type: none"> EL2 is implemented and enabled in the current Security state. m >= UInt(AArch64-MDCR_EL2.HPMN). Accessed at EL0 or EL1. m >= UInt(AArch64-PMCR_ELO.N). Otherwise access to this field is W1C. 	31 {x}

Access

MRS <Xt>, PMCNTENCLR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b010

MSR PMCNTENCLR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b010

Accessibility

MRS <Xt>, PMCNTENCLR_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCNTENCLR_ELO;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCNTENCLR_ELO;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = PMCNTENCLR_ELO;

```

MSR PMCNTENCLR_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMCNTENCLR_ELO = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMCNTENCLR_ELO = X[t, 64];
    elsif PSTATE.EL == EL2 then
        PMCNTENCLR_ELO = X[t, 64];

```

A.2.3.22 PMOVSLR_ELO, Performance Monitors Overflow Flag Status Clear Register

Contains the state of the overflow bit for the Cycle Count Register, AArch64-PMCCNTR_ELO, and each of the implemented event counters AArch64-PMEVCNTR<n>_ELO. Writing to this register clears these bits.

Configurations

AArch64 register PMOVSLR_ELO bits [31:0] are architecturally mapped to External register [B.2.2.8.13 PMOVSLR_ELO, Performance Monitors Overflow Flag Status Clear register](#) on page 2124 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

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



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-188: AARCH64_PMOVSLR_ELO bit assignments

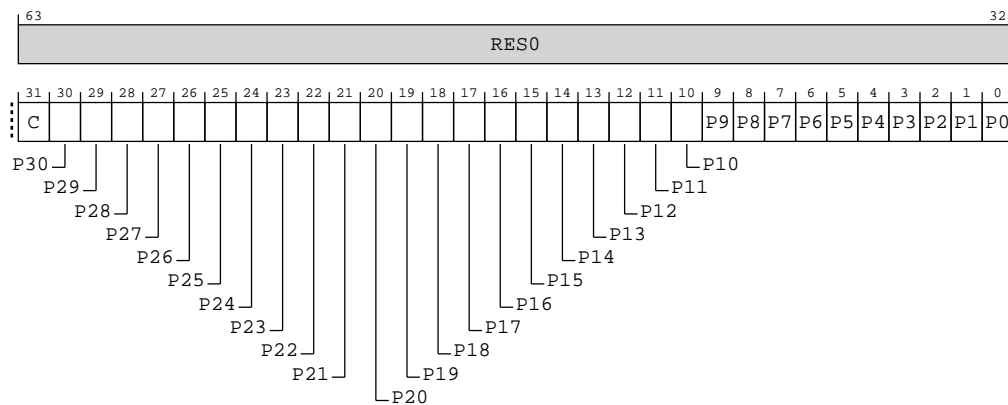


Table A-507: PMOVSLR_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	<p>Unsigned overflow flag for AArch64-PMCCNTR_ELO clear. On writes, allows software to clear the unsigned overflow flag for AArch64-PMCCNTR_ELO to 0. On reads, returns the unsigned overflow flag for AArch64-PMCCNTR_ELO overflow status.</p> <p>0b0 AArch64-PMCCNTR_ELO has not overflowed.</p> <p>0b1 AArch64-PMCCNTR_ELO has overflowed.</p> <p>Access to this field is W1C.</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Unsigned overflow flag for PMEVCNTR<m>_ELO clear. On writes, allows software to clear the unsigned overflow flag for PMEVCNTR<m>_ELO to 0. On reads, returns the unsigned overflow flag for PMEVCNTR<m>_ELO overflow status.</p> <p>0b0 PMEVCNTR<m>_ELO has not overflowed.</p> <p>0b1 PMEVCNTR<m>_ELO has overflowed.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field reads-as-zero and ignores writes if any of the following are true: <ul style="list-style-type: none"> All of the following are true: <ul style="list-style-type: none"> EL2 is implemented and enabled in the current Security state. m >= UInt(AArch64-MDCR_EL2.HPMN). Accessed at EL0 or EL1. m >= UInt(AArch64-PMCR_ELO.N). Otherwise access to this field is W1C. 	31 {x}

Access

MRS <Xt>, PMOVSLR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b011

MSR PMOVSLR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b011

Accessibility

MRS <Xt>, PMOVSLR_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);

```

```

        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMOVSLR_EL0;
        elsif PSTATE.EL == EL1 then
            if MDCR_EL2.TPM == '1' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            else
                X[t, 64] = PMOVSLR_EL0;
        elsif PSTATE.EL == EL2 then
            X[t, 64] = PMOVSLR_EL0;

```

MSR PMOVSLR_EL0, <Xt>

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMOVSLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMOVSLR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        PMOVSLR_EL0 = X[t, 64];

```

A.2.3.23 PMSWINC_EL0, Performance Monitors Software Increment Register

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see SW_INCR in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

AArch64 register PMSWINC_EL0 bits [31:0] are architecturally mapped to External register [B.2.2.8.14 PMSWINC_EL0, Performance Monitors Software Increment Register](#) on page 2126 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

```

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

```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-189: AARCH64_PMSWINC_ELO bit assignments

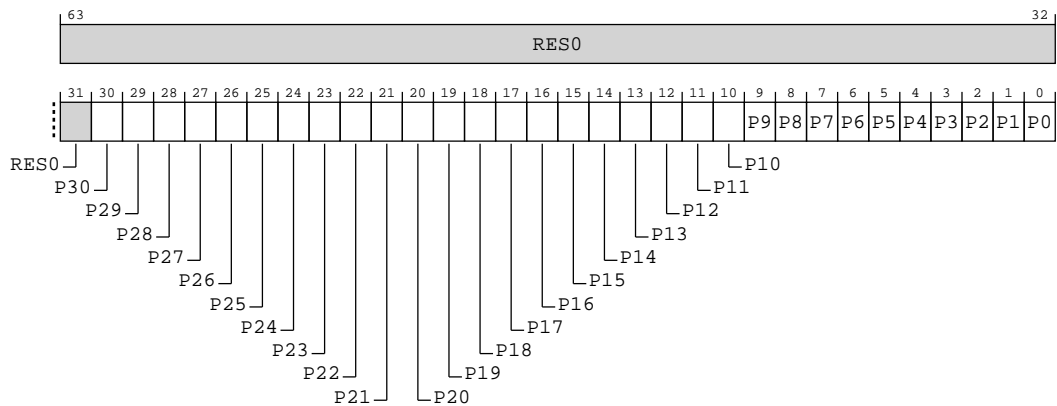


Table A-510: PMSWINC_ELO bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Software increment.</p> <p>0b0</p> <p>Write is ignored.</p> <p>0b1</p> <p>Increment PMEVCNTR<m>_ELO, if PMEVCNTR<m>_ELO is configured to count software increment events.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field ignores writes if any of the following are true: <ul style="list-style-type: none"> All of the following are true: <ul style="list-style-type: none"> EL2 is implemented and enabled in the current Security state. m >= UInt(AArch64-MDCR_EL2.HPMN). Accessed at ELO or EL1. m >= UInt(AArch64-PMCR_ELO.N). Otherwise access to this field is write-only. 	31 {x}

Access

MSR PMSWINC_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b100

Accessibility

MSR PMSWINC_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<SW,EN> == '00' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMSWINC_ELO = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMSWINC_ELO = X[t, 64];
    elsif PSTATE.EL == EL2 then
        PMSWINC_ELO = X[t, 64];

```

A.2.3.24 PMSELR_ELO, Performance Monitors Event Counter Selection Register

Selects the current event counter PMEVCNTR<n>_EL1 or the cycle counter AArch32-PMCCNTR.

Used in conjunction with AArch64-PMXEVTYPER_ELO to determine the event that increments a selected counter, and the modes and states in which the selected counter increments.

Used in conjunction with AArch64-PMXEVCNTR_ELO to determine the value of a selected counter.

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-190: AARCH64_PMSELR_ELO bit assignments

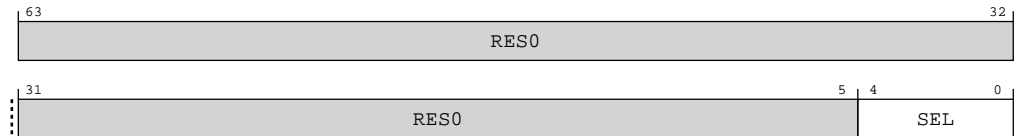


Table A-512: PMSELR_ELO bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	RES0
[4:0]	SEL	Event counter select. Selects the counter accessed by subsequent accesses to AArch64-PMXEVTYPYPER_ELO and AArch64-PMXEVCNTR_ELO. For more information about the results of accesses to the event counters, including when PMSELR_ELO.SEL is set to the index of an unimplemented or inaccessible event counter, see AArch64-PMXEVTYPYPER_ELO and AArch64-PMXEVCNTR_ELO.	5 {x}

Access

MRS <Xt>, PMSELR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b101

MSR PMSELR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b101

Accessibility

MRS <Xt>, PMSELR_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMSELR_ELO;
    elsif PSTATE.EL == EL1 then

```

```

    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMSELR_EL0;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = PMSELR_EL0;

```

MSR PMSELR_EL0, <Xt>

```

    if PSTATE.EL == EL0 then
        if PMUSERENR_EL0.<ER,EN> == '00' then
            if HCR_EL2.TGE == '1' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            else
                AArch64.SystemAccessTrap(EL1, 0x18);
            elsif MDCR_EL2.TPM == '1' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            else
                PMSELR_EL0 = X[t, 64];
        elsif PSTATE.EL == EL1 then
            if MDCR_EL2.TPM == '1' then
                AArch64.SystemAccessTrap(EL2, 0x18);
            else
                PMSELR_EL0 = X[t, 64];
        elsif PSTATE.EL == EL2 then
            PMSELR_EL0 = X[t, 64];

```

A.2.3.25 PMCEID0_ELO, 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>_ELO registers see *The PMU event number space and common events* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

AArch64 register PMCEID0_ELO bits [31:0] are architecturally mapped to External register [B.2.2.8.18 PMCEID0, Performance Monitors Common Event Identification register 0](#) on page 2134 bits [31:0].

AArch64 register PMCEID0_ELO bits [63:32] are architecturally mapped to External register [B.2.2.8.20 PMCEID2, Performance Monitors Common Event Identification register 2](#) on page 2144 bits [31:0].

Attributes

Width

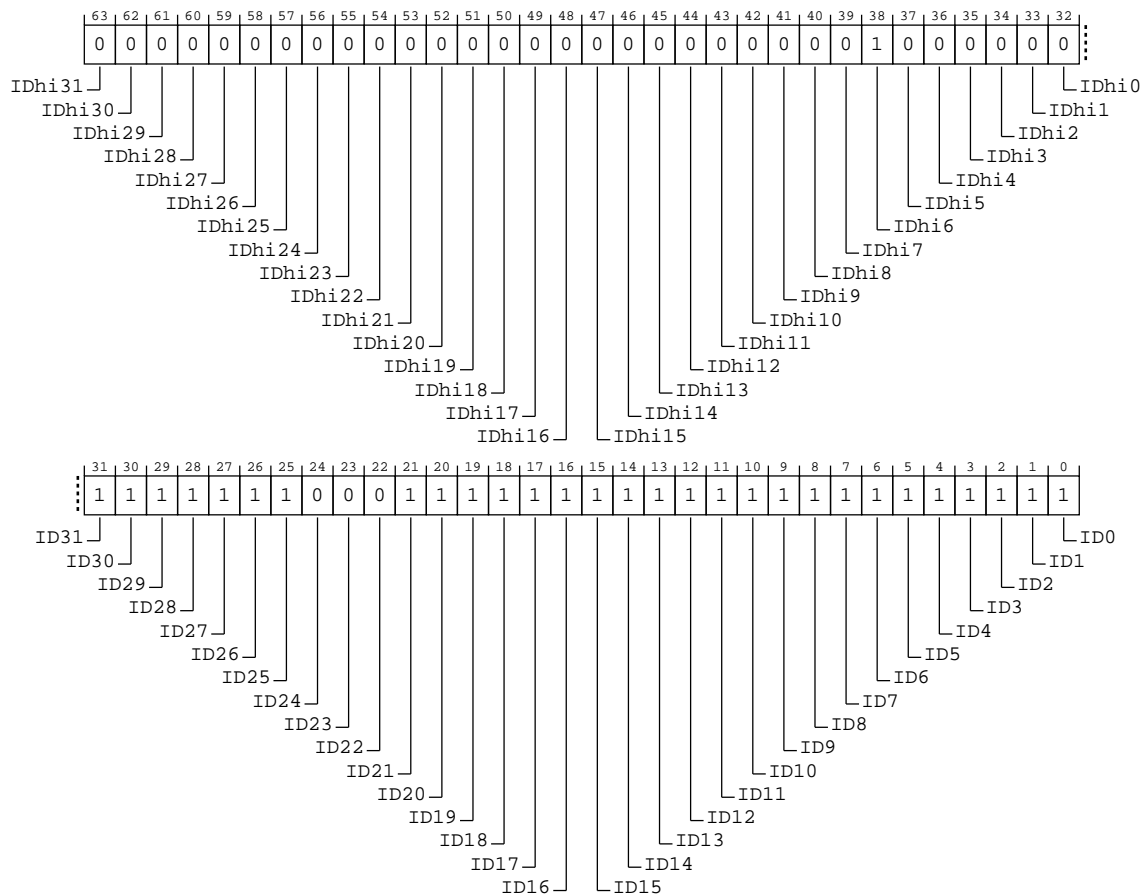
64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value0000 0000 0000 0000 0000 0000 0100 0000 1111 1110 0011 1111 1111 1111 1111
1111**Bit descriptions****Figure A-191: AARCH64_PMCEID0_ELO bit assignments****Table A-515: PMCEID0_ELO bit descriptions**

Bits	Name	Description	Reset
[63]	IDhi31	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401F not implemented.	0b0

Bits	Name	Description	Reset
[62]	IDHi30	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401E not implemented.	0b0
[61]	IDHi29	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401D not implemented.	0b0
[60]	IDHi28	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401C not implemented.	0b0
[59]	IDHi27	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401B not implemented.	0b0
[58]	IDHi26	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401A not implemented.	0b0
[57]	IDHi25	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4019 not implemented.	0b0
[56]	IDHi24	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4018 not implemented.	0b0
[55]	IDHi23	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4017 not implemented.	0b0
[54]	IDHi22	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4016 not implemented.	0b0

Bits	Name	Description	Reset
[53]	IDHi21	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4015 not implemented.	0b0
[52]	IDHi20	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4014 not implemented.	0b0
[51]	IDHi19	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4013 not implemented.	0b0
[50]	IDHi18	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4012 not implemented.	0b0
[49]	IDHi17	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4011 not implemented.	0b0
[48]	IDHi16	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4010 not implemented.	0b0
[47]	IDHi15	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400F not implemented.	0b0
[46]	IDHi14	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400E not implemented.	0b0
[45]	IDHi13	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400D not implemented.	0b0

Bits	Name	Description	Reset
[44]	IDHi12	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400C not implemented.	0b0
[43]	IDHi11	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400B not implemented.	0b0
[42]	IDHi10	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400A not implemented.	0b0
[41]	IDHi9	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4009 not implemented.	0b0
[40]	IDHi8	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4008 not implemented.	0b0
[39]	IDHi7	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4007 not implemented.	0b0
[38]	IDHi6	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b1 L1I_CACHE_LMISS event implemented.	0b1
[37]	IDHi5	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4005 not implemented.	0b0
[36]	IDHi4	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4004 not implemented.	0b0

Bits	Name	Description	Reset
[35]	IDHi3	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4003 not implemented.	0b0
[34]	IDHi2	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4002 not implemented.	0b0
[33]	IDHi1	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4001 not implemented.	0b0
[32]	IDHi0	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4000 not implemented.	0b0
[31]	ID31	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE_ALLOCATE event implemented.	0b1
[30]	ID30	ID[n] corresponds to Common event n. For each bit: 0b1 CHAIN event implemented.	0b1
[29]	ID29	ID[n] corresponds to Common event n. For each bit: 0b1 BUS_CYCLES event implemented.	0b1
[28]	ID28	ID[n] corresponds to Common event n. For each bit: 0b1 TTBR_WRITE_RETIRED event implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event n. For each bit: 0b1 INST_SPEC event implemented.	0b1

Bits	Name	Description	Reset
[26]	ID26	ID[n] corresponds to Common event n. For each bit: 0b1 MEMORY_ERROR event implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event n. For each bit: 0b1 BUS_ACCESS event implemented.	0b1
[24]	ID24	ID[n] corresponds to Common event n. For each bit: 0b0 Event 0x0018 not implemented.	0b0
[23]	ID23	ID[n] corresponds to Common event n. For each bit: 0b0 Event 0x0017 not implemented.	0b0
[22]	ID22	ID[n] corresponds to Common event n. For each bit: 0b0 Event 0x0016 not implemented.	0b0
[21]	ID21	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE_WB event implemented.	0b1
[20]	ID20	ID[n] corresponds to Common event n. For each bit: 0b1 L1I_CACHE event implemented.	0b1
[19]	ID19	ID[n] corresponds to Common event n. For each bit: 0b1 MEM_ACCESS event implemented.	0b1
[18]	ID18	ID[n] corresponds to Common event n. For each bit: 0b1 BR_PRED event implemented.	0b1

Bits	Name	Description	Reset
[17]	ID17	ID[n] corresponds to Common event n. For each bit: 0b1 CPU_CYCLES event implemented.	0b1
[16]	ID16	ID[n] corresponds to Common event n. For each bit: 0b1 BR_MIS_PRED event implemented.	0b1
[15]	ID15	ID[n] corresponds to Common event n. For each bit: 0b1 UNALIGNED_LDST_RETIRED event implemented.	0b1
[14]	ID14	ID[n] corresponds to Common event n. For each bit: 0b1 BR_RETURN_RETIRED event implemented.	0b1
[13]	ID13	ID[n] corresponds to Common event n. For each bit: 0b1 BR_IMMED_RETIRED event implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event n. For each bit: 0b1 PC_WRITE_RETIRED event implemented.	0b1
[11]	ID11	ID[n] corresponds to Common event n. For each bit: 0b1 CID_WRITE_RETIRED event implemented.	0b1
[10]	ID10	ID[n] corresponds to Common event n. For each bit: 0b1 EXC_RETURN event implemented.	0b1
[9]	ID9	ID[n] corresponds to Common event n. For each bit: 0b1 EXC_TAKEN event implemented.	0b1

Bits	Name	Description	Reset
[8]	ID8	ID[n] corresponds to Common event n. For each bit: 0b1 INST_RETIRED event implemented.	0b1
[7]	ID7	ID[n] corresponds to Common event n. For each bit: 0b1 ST_RETIRED event implemented.	0b1
[6]	ID6	ID[n] corresponds to Common event n. For each bit: 0b1 LD_RETIRED event implemented.	0b1
[5]	ID5	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_TLB_REFILL event implemented.	0b1
[4]	ID4	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE event implemented.	0b1
[3]	ID3	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE_REFILL event implemented.	0b1
[2]	ID2	ID[n] corresponds to Common event n. For each bit: 0b1 L1I_TLB_REFILL event implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event n. For each bit: 0b1 L1I_CACHE_REFILL event implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event n. For each bit: 0b1 SW_INCR event implemented.	0b1

Access

MRS <Xt>, PMCEID0_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b110

Accessibility

MRS <Xt>, PMCEID0_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCEID0_ELO;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCEID0_ELO;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = PMCEID0_ELO;

```

A.2.3.26 PMCEID1_ELO, 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>_ELO registers see *The PMU event number space and common events* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

AArch64 register PMCEID1_ELO bits [31:0] are architecturally mapped to External register [B.2.2.8.19 PMCEID1, Performance Monitors Common Event Identification register 1](#) on page 2139 bits [31:0].

AArch64 register PMCEID1_ELO bits [63:32] are architecturally mapped to External register [B.2.2.8.21 PMCEID3, Performance Monitors Common Event Identification register 3](#) on page 2149 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0111 1111 1111 1100 0000 1010 0000 0111 1111

Bit descriptions

Figure A-192: AARCH64_PMCEID1_ELO bit assignments

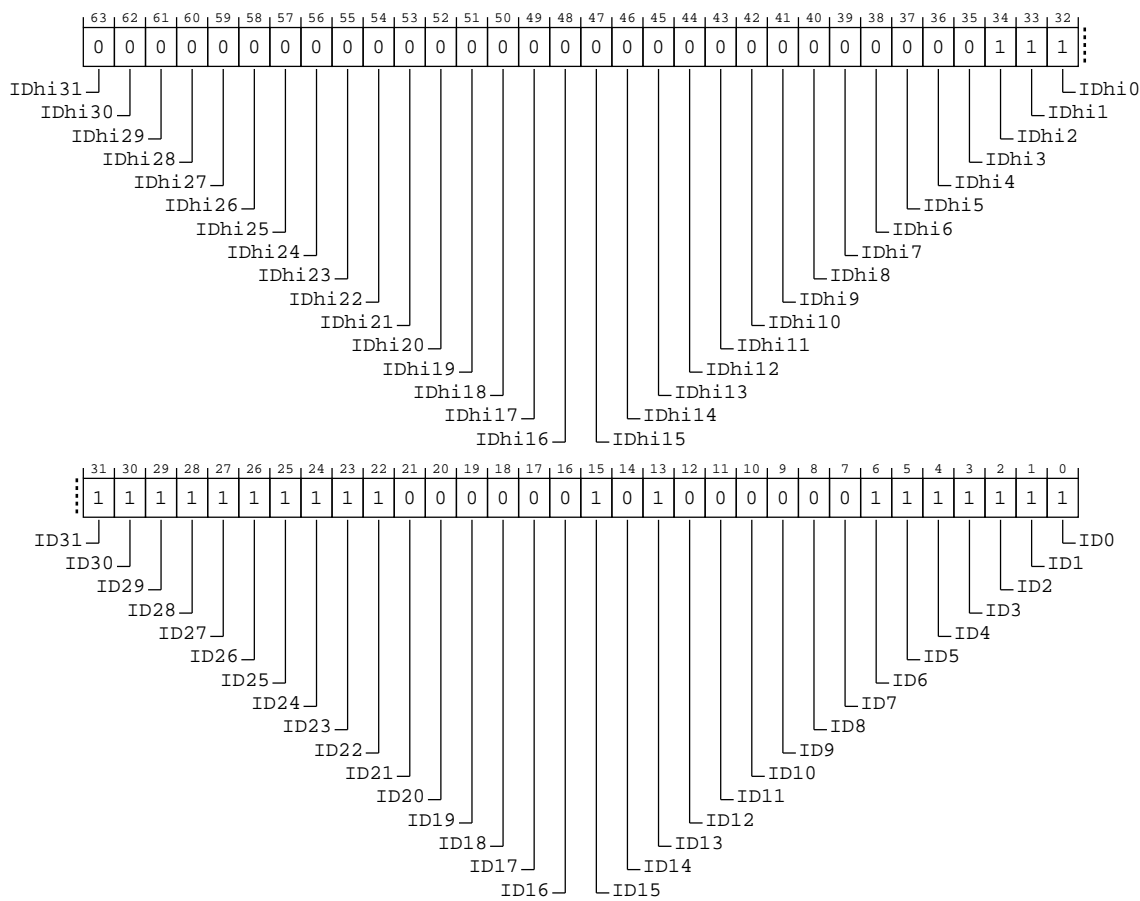


Table A-517: PMCEID1_ELO bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403F not implemented.	0b0

Bits	Name	Description	Reset
[62]	IDHi30	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403E not implemented.	0b0
[61]	IDHi29	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403D not implemented.	0b0
[60]	IDHi28	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403C not implemented.	0b0
[59]	IDHi27	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403B not implemented.	0b0
[58]	IDHi26	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403A not implemented.	0b0
[57]	IDHi25	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4039 not implemented.	0b0
[56]	IDHi24	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4038 not implemented.	0b0
[55]	IDHi23	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4037 not implemented.	0b0
[54]	IDHi22	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4036 not implemented.	0b0

Bits	Name	Description	Reset
[53]	IDHi21	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4035 not implemented.	0b0
[52]	IDHi20	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4034 not implemented.	0b0
[51]	IDHi19	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4033 not implemented.	0b0
[50]	IDHi18	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4032 not implemented.	0b0
[49]	IDHi17	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4031 not implemented.	0b0
[48]	IDHi16	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4030 not implemented.	0b0
[47]	IDHi15	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402F not implemented.	0b0
[46]	IDHi14	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402E not implemented.	0b0
[45]	IDHi13	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402D not implemented.	0b0

Bits	Name	Description	Reset
[44]	IDHi12	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402C not implemented.	0b0
[43]	IDHi11	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402B not implemented.	0b0
[42]	IDHi10	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402A not implemented.	0b0
[41]	IDHi9	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4029 not implemented.	0b0
[40]	IDHi8	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4028 not implemented.	0b0
[39]	IDHi7	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4027 not implemented.	0b0
[38]	IDHi6	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4026 not implemented.	0b0
[37]	IDHi5	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4025 not implemented.	0b0
[36]	IDHi4	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4024 not implemented.	0b0

Bits	Name	Description	Reset
[35]	IDhi3	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4023 not implemented.	0b0
[34]	IDhi2	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b1 ST_ALIGN_LAT event implemented.	0b1
[33]	IDhi1	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b1 LD_ALIGN_LAT event implemented.	0b1
[32]	IDhi0	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b1 LDST_ALIGN_LAT event implemented.	0b1
[31]	ID31	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_SLOT event implemented.	0b1
[30]	ID30	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_SLOT_FRONTEND event implemented.	0b1
[29]	ID29	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_SLOT_BACKEND event implemented.	0b1
[28]	ID28	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL event implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 OP_SPEC event implemented.	0b1

Bits	Name	Description	Reset
[26]	ID26	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 OP_RETIRED event implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L1D_CACHE_LMISS_RD event implemented.	0b1
[24]	ID24	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 REMOTE_ACCESS_RD event implemented.	0b1
[23]	ID23	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 LL_CACHE_MISS_RD event implemented.	0b1
[22]	ID22	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 LL_CACHE_RD event implemented.	0b1
[21]	ID21	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0035 not implemented.	0b0
[20]	ID20	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0034 not implemented.	0b0
[19]	ID19	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0033 not implemented.	0b0
[18]	ID18	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0032 not implemented.	0b0

Bits	Name	Description	Reset
[17]	ID17	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0031 not implemented.	0b0
[16]	ID16	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0030 not implemented.	0b0
[15]	ID15	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L2D_TLB event implemented.	0b1
[14]	ID14	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002E not implemented.	0b0
[13]	ID13	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L2D_TLB_REFILL event implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002C not implemented.	0b0
[11]	ID11	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002B not implemented.	0b0
[10]	ID10	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002A not implemented.	0b0
[9]	ID9	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0029 not implemented.	0b0

Bits	Name	Description	Reset
[8]	ID8	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0028 not implemented.	0b0
[7]	ID7	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0027 not implemented.	0b0
[6]	ID6	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L1I_TLB event implemented.	0b1
[5]	ID5	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L1D_TLB event implemented.	0b1
[4]	ID4	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_BACKEND event implemented.	0b1
[3]	ID3	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_FRONTEND event implemented.	0b1
[2]	ID2	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 BR_MIS_PRED_RETIRED event implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 BR_RETIRED event implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L2D_CACHE_ALLOCATE event implemented.	0b1

Access

MRS <Xt>, PMCEID1_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b111

Accessibility

MRS <Xt>, PMCEID1_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCEID1_ELO;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCEID1_ELO;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = PMCEID1_ELO;

```

A.2.3.27 PMCCNTR_ELO, Performance Monitors Cycle Count Register

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See *Time as measured by the Performance Monitors cycle counter* in the [Arm® Architecture Reference Manual for A-profile architecture](#) for more information.

AArch64-PMCCFILTR_ELO determines the modes and states in which the PMCCNTR_ELO can increment.

Configurations

All counters are subject to any changes in clock frequency, including clock stopping caused by the WFI and WFE instructions.

AArch64 register PMCCNTR_ELO bits [63:0] are architecturally mapped to External register [B.2.2.8.2 PMCCNTR_ELO, Performance Monitors Cycle Counter](#) on page 2105 bits [63:0].

Attributes

Width

64

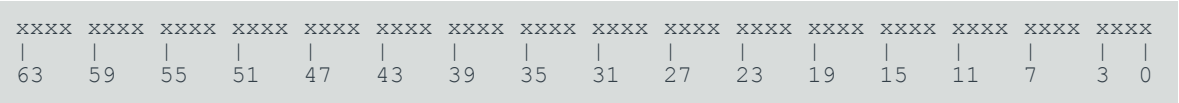
Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-193: AARCH64_PMCCNTR_ELO bit assignments

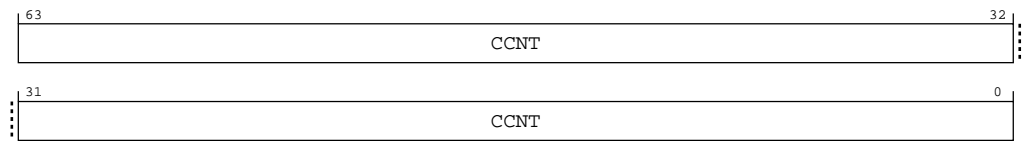


Table A-519: PMCCNTR_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	CCNT	Cycle count. Depending on the values of AArch64-PMCR_ELO.{LC,D}, this field increments in one of the following ways: <ul style="list-style-type: none">Every processor clock cycle.Every 64th processor clock cycle. Writing 1 to AArch64-PMCR_ELO.C sets this field to 0.	64 {x}

Access

MRS <Xt>, PMCCNTR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b000

MSR PMCCNTR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b000

Accessibility

MRS <Xt>, PMCCNTR_ELO

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

```

if PMUSERENR_EL0.<CR,EN> == '00' then
    if HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif MDCR_EL2.TPM == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = PMCCNTR_EL0;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMCCNTR_EL0;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PMCCNTR_EL0;

```

MSR PMCCNTR_EL0, <Xt>

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMCCNTR_EL0 = X[t, 64];
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMCCNTR_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    PMCCNTR_EL0 = X[t, 64];

```

A.2.3.28 PMXEVTYPER_EL0, Performance Monitors Selected Event Type Register

When AArch64-PMSELR_EL0.SEL selects an event counter, this accesses a AArch64-PMEVTYPER<n>_EL0 register. When AArch64-PMSELR_EL0.SEL selects the cycle counter, this accesses AArch64-PMCCFILTR_EL0.

Configurations

This register is available in all configurations.

Attributes

Width

64

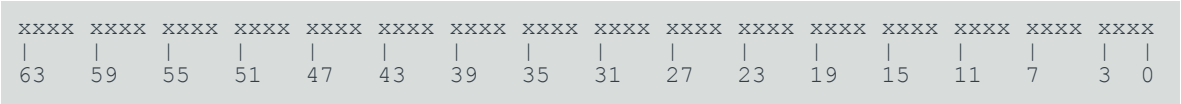
Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-194: AArch64_PMXEVTYPER_ELO bit assignments

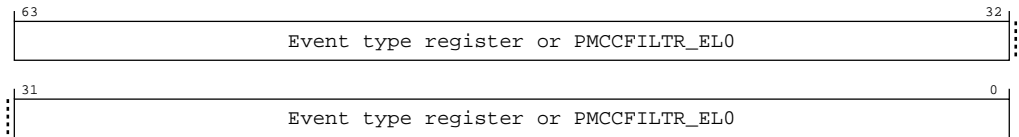


Table A-522: PMXEVTYPER_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	None	When AArch64-PMSELR_ELO.SEL == 31, this register accesses AArch64-PMCCFILTR_ELO. Otherwise, this register accesses AArch64-PMEVTYPER<n>_ELO where n is the value in AArch64-PMSELR_ELO.SEL.	64 {x}

Access

If AArch64-PMSELR_ELO.SEL is not 31, and is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMXEVTYPER_ELO are **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is: accesses to the register behave as RAZ/WI.

In ELO, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and ELO, 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>, PMXEVTYPER_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b001

MSR PMXEVTYPER_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b001

Accessibility

If AArch64-PMSELR_ELO.SEL is not 31, and is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMXEVTYPER_ELO are CONSTRAINED UNPREDICTABLE, and the implemented behavior is: accesses to the register behave as RAZ/WI.

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO.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>, PMXEVTYPER_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_ELO.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMXEVTYPER_ELO;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMXEVTYPER_ELO;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PMXEVTYPER_ELO;

```

MSR PMXEVTYPER_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if PMUSERENR_ELO.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMXEVTYPER_ELO = X[t, 64];
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMXEVTYPER_ELO = X[t, 64];
elseif PSTATE.EL == EL2 then
    PMXEVTYPER_ELO = X[t, 64];

```

A.2.3.29 PMXEVCNTR_ELO, Performance Monitors Selected Event Count Register

Reads or writes the value of the selected event counter, AArch64-PMXEVCNTR<n>_ELO. AArch64-PMSELR_ELO.SEL determines which event counter is selected.

Configurations

This register is available in all configurations.

Attributes

Width

64

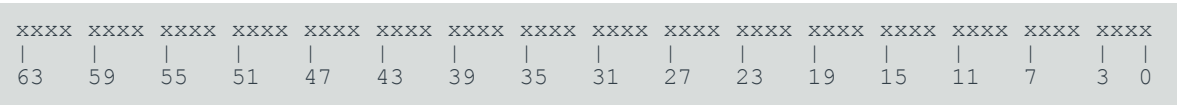
Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-195: AARCH64_PMXEVCNTR_ELO bit assignments

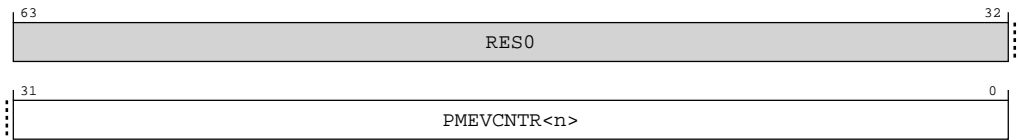


Table A-525: PMXEVCNTR_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	PMXEVCNTR<n>	Value of the selected event counter, AArch64-PMXEVCNTR<n>_ELO, where n is the value stored in AArch64-PMSELR_ELO.SEL.	32 {x}

Access

If AArch64-PMSELR_ELO.SEL is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMXEVNTR_ELO are **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is: accesses to the register behave as RAZ/WI.

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO.{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>, PMXEVNTR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b010

MSR PMXEVNTR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	0b010

Accessibility

If AArch64-PMSELR_ELO.SEL is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMXEVNTR_ELO are **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is: accesses to the register behave as RAZ/WI.

In EL0, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO.{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>, PMXEVNTR_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_ELO.<ER,EN> == '00' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMXEVNTR_ELO;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMXEVNTR_ELO;
elseif PSTATE.EL == EL2 then

```

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

MSR PMXEVCNTR_ELO, <Xt>

```
if PSTATE.EL == EL0 then
    if PMUSERENR_ELO.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMXEVCNTR_ELO = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMXEVCNTR_ELO = X[t, 64];
    elsif PSTATE.EL == EL2 then
        PMXEVCNTR_ELO = X[t, 64];
```

A.2.3.30 PMUSERENR_ELO, Performance Monitors User Enable Register

Enables or disables ELO access to the Performance Monitors.

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-196: AARCH64_PMUSERENR_ELO bit assignments

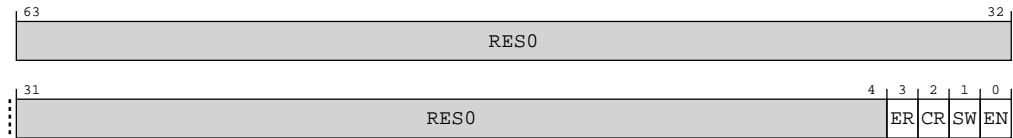


Table A-528: PMUSERENR_ELO bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3]	ER	<p>Event counters Read enable.</p> <p>When PMUSERENR_ELO.EN is 0, PMUSERENR_ELO.ER enables ELO reads of the event counters and ELO reads and writes of the select register.</p> <p>0b0</p> <p>ELO reads of the event counters and ELO reads and writes of the select register are disabled, unless enabled by PMUSERENR_ELO.EN.</p> <p>0b1</p> <p>ELO reads of the event counters and ELO reads and writes of the select register are enabled, unless trapped by another control.</p> <p>In AArch64 state, the register accesses affected by this control are:</p> <ul style="list-style-type: none"> MRS reads of AArch64-PMEVCNTR<n>_ELO and AArch64-PMXVCNTR_ELO. MRS and MSR accesses to AArch64-PMSELR_ELO. <p>When disabled, reads and writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, and:</p> <ul style="list-style-type: none"> AArch64 MRS and MSR accesses are reported using EC syndrome value 0x18. <p>This field is ignored by the PE when PMUSERENR_ELO.EN == 1.</p>	x
[2]	CR	<p>Cycle counter Read enable.</p> <p>When PMUSERENR_ELO.EN is 0, PMUSERENR_ELO.CR enables ELO reads of the cycle counter.</p> <p>0b0</p> <p>ELO reads of the cycle counter are disabled, unless enabled by PMUSERENR_ELO.EN.</p> <p>0b1</p> <p>ELO reads of the cycle counter are enabled, unless trapped by another control.</p> <p>In AArch64 state, the register accesses affected by this control are:</p> <ul style="list-style-type: none"> MRS reads of AArch64-PMCCNTR_ELO. <p>When disabled, reads generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, and:</p> <ul style="list-style-type: none"> AArch64 MRS reads are reported using EC syndrome value 0x18. <p>This field is ignored by the PE when PMUSERENR_ELO.EN == 1.</p>	x

Bits	Name	Description	Reset
[1]	SW	<p>Software increment register Write enable.</p> <p>When PMUSERENR_ELO.EN is 0, PMUSERENR_ELO.SW enables ELO writes to the Software increment register.</p> <p>0b0</p> <p>ELO writes to the Software increment register are disabled, unless enabled by PMUSERENR_ELO.EN.</p> <p>0b1</p> <p>ELO writes to the Software increment register are enabled, unless trapped by another control.</p> <p>In AArch64 state, the register accesses affected by this control are:</p> <ul style="list-style-type: none"> MSR writes to AArch64-PMSWINC_ELO. <p>When disabled, writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, and:</p> <ul style="list-style-type: none"> AArch64 MSR writes are reported using EC syndrome value 0x18. <p>This field is ignored by the PE when PMUSERENR_ELO.EN == 1.</p>	x
[0]	EN	<p>Enable.</p> <p>Enables ELO read/write access to PMU registers.</p> <p>0b0</p> <p>ELO accesses to the specified PMU System registers are trapped, unless enabled by PMUSERENR_ELO. {ER,CR,SW}.</p> <p>0b1</p> <p>ELO accesses to the specified PMU System registers are enabled, unless trapped by another control.</p> <p>In AArch64 state, the register accesses affected by this control are:</p> <ul style="list-style-type: none"> MRS or MSR accesses to AArch64-PMCCFILTR_ELO, AArch64-PMCCNTR_ELO, AArch64-PMCNTENCLR_ELO, AArch64-PMCNTENSET_ELO, AArch64-PMCR_ELO, AArch64-PMEVCNTR<n>_ELO, AArch64-PMEVTYPER<n>_ELO, AArch64-PMOVSCLR_ELO, AArch64-PMOVSSET_ELO, AArch64-PMSELR_ELO, AArch64-PMXEVCNTR_ELO, and AArch64-PMXEVTYPER_ELO. MRS reads of AArch64-PMCEIDO_ELO and AArch64-PMCEID1_ELO. MSR writes to AArch64-PMSWINC_ELO. <p>When trapped, reads and writes generate an exception to EL1, or to EL2 when EL2 is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, and:</p> <ul style="list-style-type: none"> AArch64 MRS and MSR accesses are reported using EC syndrome value 0x18. 	x

Access

MRS <Xt>, PMUSERENR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b000

MSR PMUSERENR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b000

Accessibility

MRS <Xt>, PMUSERENR_ELO

```
if PSTATE.EL == EL0 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMUSERENR_ELO;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PMUSERENR_ELO;
elseif PSTATE.EL == EL2 then
    X[t, 64] = PMUSERENR_ELO;
```

MSR PMUSERENR_ELO, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMUSERENR_ELO = X[t, 64];
elseif PSTATE.EL == EL2 then
    PMUSERENR_ELO = X[t, 64];
```

A.2.3.31 PMOVSET_ELO, Performance Monitors Overflow Flag Status Set Register

Sets the state of the overflow bit for the Cycle Count Register, AArch64-PMCCNTR_ELO, and each of the implemented event counters AArch64-PMEVCNTR<n>_ELO.

Configurations

AArch64 register PMOVSET_ELO bits [31:0] are architecturally mapped to External register [B.2.2.8.15 PMOVSET_ELO, Performance Monitors Overflow Flag Status Set Register](#) on page 2128 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers


Access type

See bit descriptions

Reset value



6359555147433935312723191511730



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-197: AArch64_PMOVSET_ELO bit assignments

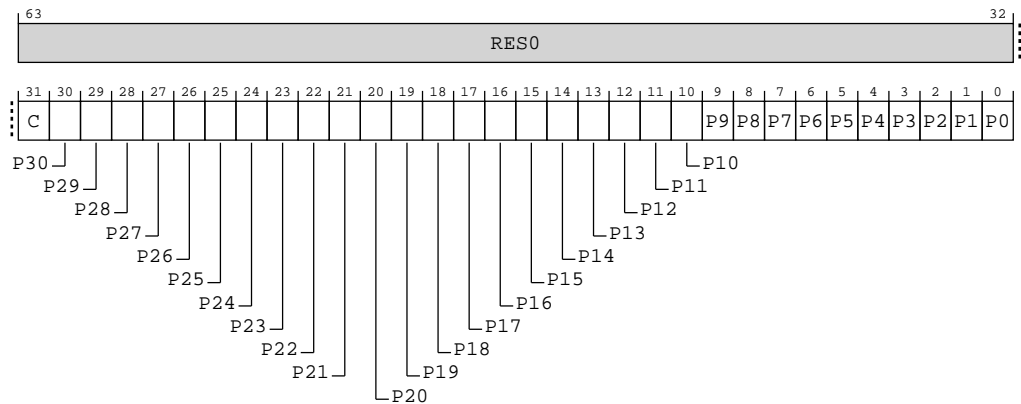


Table A-531: PMOVSET_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	C	Unsigned overflow flag for AArch64-PMCCNTR_ELO set. On writes, allows software to set the unsigned overflow flag for AArch64-PMCCNTR_ELO to 1. On reads, returns the unsigned overflow flag for AArch64-PMCCNTR_ELO overflow status. 0b0 AArch64-PMCCNTR_ELO has not overflowed. 0b1 AArch64-PMCCNTR_ELO has overflowed. Access to this field is W1C.	x

Bits	Name	Description	Reset
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Unsigned overflow flag for PMEVCNTR<m>_ELO set. On writes, allows software to set the unsigned overflow flag for PMEVCNTR<m>_ELO to 1. On reads, returns the unsigned overflow flag for PMEVCNTR<m>_ELO overflow status.</p> <p>0b0</p> <p>PMEVCNTR<m>_ELO has not overflowed.</p> <p>0b1</p> <p>PMEVCNTR<m>_ELO has overflowed.</p> <p>Accessing this field has the following behavior:</p> <ul style="list-style-type: none"> This field reads-as-zero and ignores writes if any of the following are true: <ul style="list-style-type: none"> All of the following are true: <ul style="list-style-type: none"> EL2 is implemented and enabled in the current Security state. m >= UInt(AArch64-MDCR_EL2.HPMN). Accessed at EL0 or EL1. m >= UInt(AArch64-PMCR_ELO.N). Otherwise access to this field is W1S. 	31 {x}

Access

MRS <Xt>, PMOVSSET_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b011

MSR PMOVSSET_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1110	0b011

Accessibility

MRS <Xt>, PMOVSSET_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMOVSSET_ELO;
    elseif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMOVSSET_ELO;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = PMOVSSET_ELO;

```

MSR PMOVSSET_ELO, <Xt>

```
if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMOVSSET_ELO = X[t, 64];
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            PMOVSSET_ELO = X[t, 64];
    elsif PSTATE.EL == EL2 then
        PMOVSSET_ELO = X[t, 64];
```

A.2.3.32 PMEVCNTR<n>_ELO, Performance Monitors Event Count Registers, n = 0 - 5

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

AArch64 register PMEVCNTR<n>_ELO bits [31:0] are architecturally mapped to External register [B.2.2.8.1 PMEVCNTR<n>_ELO, Performance Monitors Event Count Registers, n = 0 - 5](#) on page 2104 bits [31:0].

Attributes

Width

64

Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-198: AARCH64_PMEVCNTR<n>_ELO bit assignments

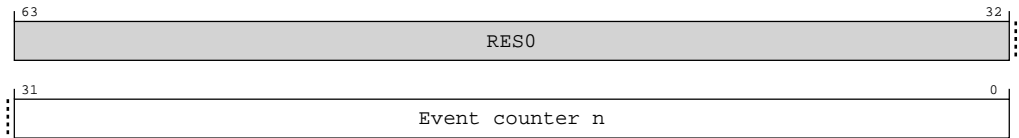


Table A-534: PMEVCNTR<n>_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	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.	32 {x}

Access

PMEVCNTR<n>_ELO can also be accessed by using AArch64-PMXEVCNTR_ELO with AArch64-PMSELR_ELO.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVCNTR<n>_ELO are **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is: accesses to the register behave as RAZ/WI.



In ELO, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO. {ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and ELO, 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>, PMEVCNTR<m>_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	'10':m[4:3]	m[2:0]

MSR PMEVCNTR<m>_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	'10':m[4:3]	m[2:0]

Accessibility

PMEVCNTR<n>_ELO can also be accessed by using AArch64-PMXEVCNTR_ELO with AArch64-PMSELR_ELO.SEL set to the value of <n>.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMUVCNTR<n>_ELO are CONSTRAINED UNPREDICTABLE, and the implemented behavior is: accesses to the register behave as RAZ/WI.



In ELO, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO.
{ER,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and ELO, 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>, PMUVCNTR<m>_ELO

```
integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= 6 then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elseif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.<ER,EN> == '00' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif m >= AArch64.GetNumEventCountersAccessible() then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            X[t, 64] = PMUVCNTR_EL0[m];
    elseif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif m >= AArch64.GetNumEventCountersAccessible() then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            X[t, 64] = PMUVCNTR_EL0[m];
    elseif PSTATE.EL == EL2 then
        X[t, 64] = PMUVCNTR_EL0[m];
```

MSR PMUVCNTR<m>_ELO, <Xt>

```
integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= 6 then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elseif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif m >= AArch64.GetNumEventCountersAccessible() then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            PMUVCNTR_EL0[m] = X[t, 64];
    elseif PSTATE.EL == EL1 then
```

```
if MDCR_EL2.TPM == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif m >= AArch64.GetNumEventCountersAccessible() then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
else
    PMEVCNTR_EL0[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    PMEVCNTR_EL0[m] = X[t, 64];
```

A.2.3.33 PMEVTYPER<n>_ELO, Performance Monitors Event Type Registers, n = 0 - 5

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-199: AARCH64_PMEVTYPER<n>_ELO bit assignments

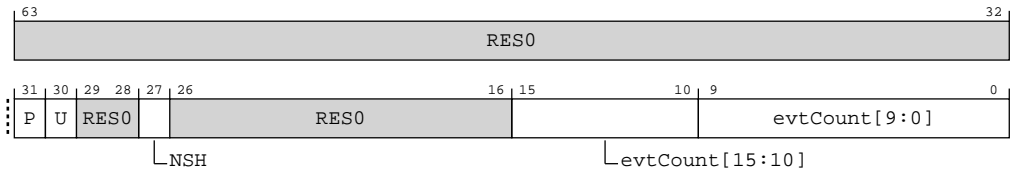


Table A-537: PMEVTYPER<n>_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	P	<p>EL1 filtering. Controls counting events in EL1.</p> <p>0b0 This field has no effect on filtering of events.</p> <p>0b1 Events in EL1 are not counted.</p>	x
[30]	U	<p>ELO filtering. Controls counting events in ELO.</p> <p>0b0 This field has no effect on filtering of events.</p> <p>0b1 Events in ELO are not counted.</p>	x
[29:28]	RES0	Reserved	RES0
[27]	NSH	<p>EL2 filtering. Controls counting events in EL2.</p> <p>0b0 Events in EL2 are not counted.</p> <p>0b1 This field has no effect on filtering of events.</p>	x
[26:16]	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]	<p>Event to count.</p> <p>The event number of the event that is counted by event counter AArch64-PMEVCNTR<n>_ELO.</p> <p>The ranges of event numbers allocated to each type of event are shown in <i>Allocation of the PMU event number space</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>If PMEVTYPER<n>_ELO.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:</p> <ul style="list-style-type: none"> For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_ELO.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>_ELO.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>_ELO.evtCount field is UNKNOWN. <p>Note: UNPREDICTABLE means the event must not expose privileged information.</p>	10 {x}

Access

PMEVTYPER<n>_ELO can also be accessed by using AArch64-PMXEVTYPER_ELO with AArch64-PMSELR_ELO.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_ELO are **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is: accesses to the register behave as RAZ/WI.



In ELO, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and ELO, 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>, PMEVTYPER<m>_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	'11':m[4:3]	m[2:0]

MSR PMEVTYPER<m>_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	'11':m[4:3]	m[2:0]

Accessibility

PMEVTYPER<n>_ELO can also be accessed by using AArch64-PMXEVTYPER_ELO with AArch64-PMSELR_ELO.SEL set to n.

If <n> is greater than or equal to the number of accessible event counters, then reads and writes of AArch64-PMEVTYPER<n>_ELO are **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is: accesses to the register behave as RAZ/WI.



In ELO, an access is permitted if it is enabled by AArch64-PMUSERENR_ELO.EN.

If EL2 is implemented and enabled in the current Security state, in EL1 and ELO, 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>, PMEVTYPER<m>_ELO

```
integer m = UInt(CRm<1:0>:op2<2:0>);
if m >= 6 then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
```

```

elseif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif m >= AArch64.GetNumEventCountersAccessible() then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            X[t, 64] = PMEVTYPER_EL0[m];
    elseif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif m >= AArch64.GetNumEventCountersAccessible() then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            X[t, 64] = PMEVTYPER_EL0[m];
    elseif PSTATE.EL == EL2 then
        X[t, 64] = PMEVTYPER_EL0[m];

```

MSR PMEVTYPER<m>_EL0, <Xt>

```

integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= 6 then
    ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elseif PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif m >= AArch64.GetNumEventCountersAccessible() then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            PMEVTYPER_EL0[m] = X[t, 64];
    elseif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif m >= AArch64.GetNumEventCountersAccessible() then
            ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            PMEVTYPER_EL0[m] = X[t, 64];
    elseif PSTATE.EL == EL2 then
        PMEVTYPER_EL0[m] = X[t, 64];

```

A.2.3.34 PMCCFILTR_EL0, Performance Monitors Cycle Count Filter Register

Determines the modes in which the Cycle Counter, AArch64-PMCCNTR_EL0, increments.

Configurations

AArch64 register PMCCFILTR_EL0 bits [31:0] are architecturally mapped to External register [B.2.2.8.8 PMCCFILTR_EL0, Performance Monitors Cycle Counter Filter Register](#) on page 2115 bits [31:0].

Attributes

Width

64

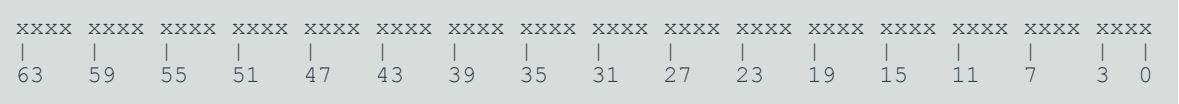
Functional group

Performance Monitors registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-200: AARCH64_PMCCFILTR_EL0 bit assignments

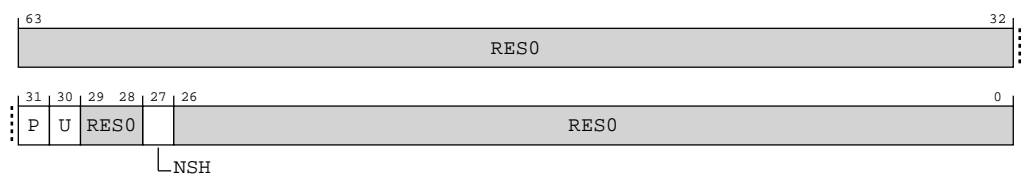


Table A-540: PMCCFILTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	P	EL1 filtering. Controls counting cycles in EL1. 0b0 This field has no effect on filtering of cycles. 0b1 Cycles in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting cycles in EL0. 0b0 This field has no effect on filtering of cycles. 0b1 Cycles in EL0 are not counted.	x
[29:28]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[27]	NSH	EL2 filtering. Controls counting cycles in EL2. 0b0 Cycles in EL2 are not counted. 0b1 This field has no effect on filtering of cycles.	x
[26:0]	RES0	Reserved	RES0

Access

PMCCFILTR_ELO can also be accessed by using AArch64-PMXEVTYPER_ELO with AArch64-PMSELR_ELO.SEL set to 0b11111.

MRS <Xt>, PMCCFILTR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1111	0b111

MSR PMCCFILTR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b1111	0b111

Accessibility

PMCCFILTR_ELO can also be accessed by using AArch64-PMXEVTYPER_ELO with AArch64-PMSELR_ELO.SEL set to 0b11111.

MRS <Xt>, PMCCFILTR_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCCFILTR_ELO;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = PMCCFILTR_ELO;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = PMCCFILTR_ELO;

```

MSR PMCCFILTR_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);

```

```
elseif MDCR_EL2.TPM == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    PMCCFILTR_EL0 = X[t, 64];
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PMCCFILTR_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    PMCCFILTR_EL0 = X[t, 64];
```

A.2.4 AArch64 System instruction register description

This section includes the register descriptions for all System instruction registers in the Cortex®-R82 processor.

A.2.4.1 IC IALLUIS, Instruction Cache Invalidate All to PoU, Inner Shareable

Invalidate all instruction caches in the Inner Shareable domain of the PE executing the instruction to the Point of Unification.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

IC IALLUIS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0001	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

IC IALLUIS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TPU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.IC(CacheOpScope_ALLUIS);
elsif PSTATE.EL == EL2 then
    AArch64.IC(CacheOpScope_ALLUIS);

```

A.2.4.2 IC IALLU, Instruction Cache Invalidate All to PoU

Invalidate all instruction caches of the PE executing the instruction to the Point of Unification.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

IC IALLU{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0101	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

IC IALLU{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TPU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FB == '1' then
        AArch64.IC(CacheOpScope_ALLUIS);

```

```
else
    AArch64.IC(CacheOpScope_ALLU);
elseif PSTATE.EL == EL2 then
    AArch64.IC(CacheOpScope_ALLU);
```

A.2.4.3 DC IVAC, Data or unified Cache line Invalidate by VA to PoC

Invalidate data cache by address to Point of Coherency.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-201: AARCH64_DC_IVAC bit assignments

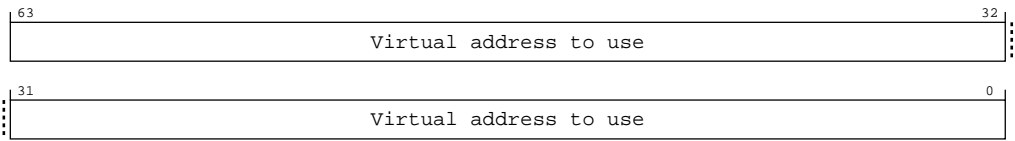


Table A-545: DC IVAC bit descriptions

Bits	Name	Description	Reset
[63:0]	VA	Virtual address to use. No alignment restrictions apply to this VA.	64 {x}

Access

When the instruction is executed, it can generate a watchpoint, which is prioritized in the same way as other watchpoints. If a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 1.

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in *MMU faults generated by cache maintenance operations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see *The data cache maintenance instruction (DC)* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

DC IVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b001

Accessibility

When the instruction is executed, it can generate a watchpoint, which is prioritized in the same way as other watchpoints. If a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 1.

If ELO access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.
DC IVAC, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TPCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_PoC);

```

A.2.4.4 DC ISW, Data or unified Cache line Invalidate by Set/Way

Invalidate data cache by set/way.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-202: AARCH64_DC_ISW bit assignments

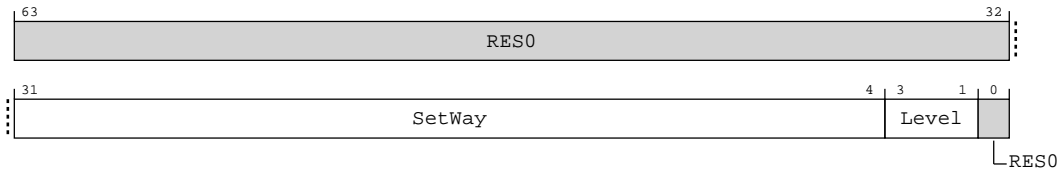


Table A-547: DC ISW bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:4]	SetWay	Contains two fields: <ul style="list-style-type: none">Way, bits[31:32-A], the number of the way to operate on.Set, bits[B-1:L], the number of the set to operate on. Bits[L-1:4] are RES0. $A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$. ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.	28 {x}
[3:1]	Level	Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.	xxx
[0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is **CONSTRAINED UNPREDICTABLE** and the implemented behavior is:

- The instruction performs cache maintenance on a single arbitrary cache line.

DC ISW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b0110	0b010

Accessibility

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is **CONSTRAINED UNPREDICTABLE** and the implemented behavior is:

- The instruction performs cache maintenance on a single arbitrary cache line.

DC ISW, <Xt>

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

```
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    if HCR_EL2.TSW == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate,
        CacheOpScope_SetWay);
    elseif PSTATE.EL == EL2 then
      AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Invalidate, CacheOpScope_SetWay);
```

A.2.4.5 AT S1E1R, Address Translate Stage 1 EL1 Read

Performs stage 1 address translation, with permissions as if reading from the given virtual address from EL1, using the EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-203: AARCH64_AT_S1E1R bit assignments

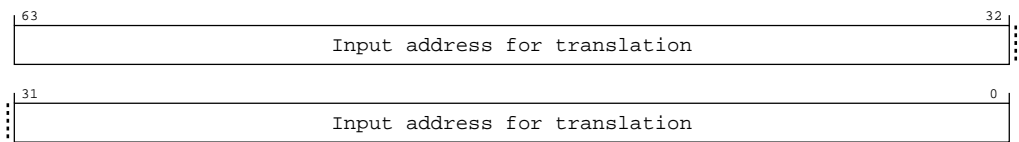


Table A-549: AT S1E1R bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S1E1R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b000

Accessibility

AT S1E1R, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Read);
elseif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Read);
```

A.2.4.6 AT S1E1W, Address Translate Stage 1 EL1 Write

Performs stage 1 address translation, with permissions as if writing to the given virtual address from EL1, using the EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-204: AARCH64_AT_S1E1W bit assignments

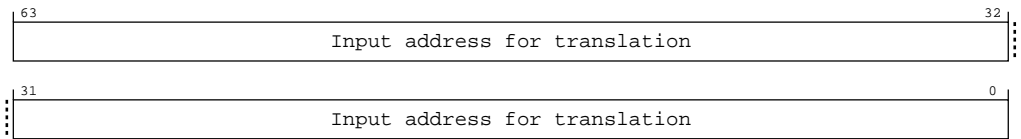


Table A-551: AT S1E1W bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S1E1W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b001

Accessibility

AT S1E1W, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Write);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Write);
```

A.2.4.7 AT S1EOR, Address Translate Stage 1 ELO Read

Performs stage 1 address translation from ELO, with permissions as if reading from the given virtual address from ELO, using the EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-205: AARCH64_AT_S1EOR bit assignments

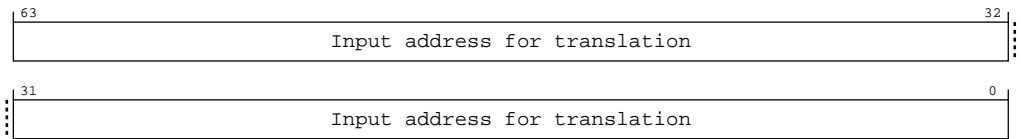


Table A-553: AT S1EOR bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S1EOR, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b010

Accessibility

AT S1EOR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Read);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Read);
```

A.2.4.8 AT S1EOW, Address Translate Stage 1 ELO Write

Performs stage 1 address translation from ELO, with permissions as if writing to the given virtual address from ELO, using the EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-206: AARCH64_AT_S1EOW bit assignments

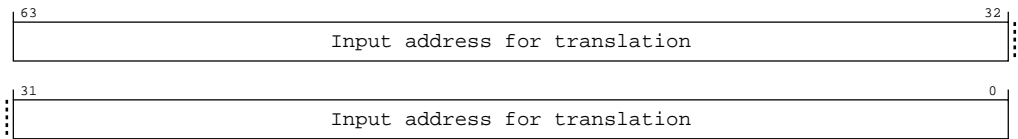


Table A-555: AT S1EOW bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S1EOW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1000	0b011

Accessibility

AT S1E0W, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Write);
elsif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Write);
```

A.2.4.9 AT S1E1RP, Address Translate Stage 1 EL1 Read PAN

Performs a stage 1 address translation, where the value of PSTATE.PAN determines if a read from a location will generate a Permission fault for a privileged access, using the EL1&0 translation regime, accessed from EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-207: AARCH64_AT_S1E1RP bit assignments

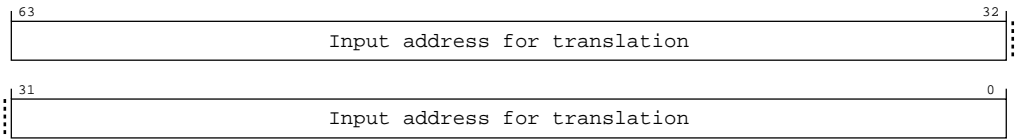


Table A-557: AT S1E1RP bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S1E1RP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1001	0b000

Accessibility

AT S1E1RP, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_ReadPAN);
elseif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_ReadPAN);
```

A.2.4.10 AT S1E1WP, Address Translate Stage 1 EL1 Write PAN

Performs a stage 1 address translation, where the value of PSTATE.PAN determines if a write to a location will generate a Permission fault for a privileged access, using the EL1&0 translation regime, accessed from EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-208: AARCH64_AT_S1E1WP bit assignments

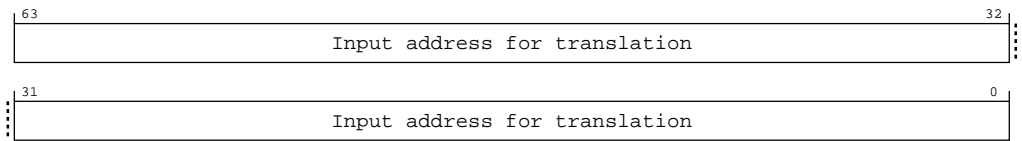


Table A-559: AT S1E1WP bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S1E1WP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1001	0b001

Accessibility

AT S1E1WP, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_WritePAN);
elseif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_WritePAN);
```

A.2.4.11 DC CSW, Data or unified Cache line Clean by Set/Way

Clean data cache by set/way.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-209: AARCH64_DC_CSW bit assignments

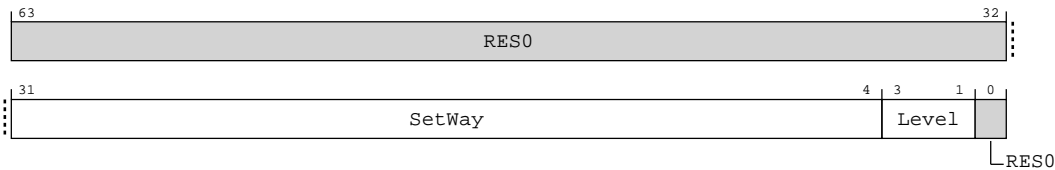


Table A-561: DC CSW bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:4]	SetWay	Contains two fields: <ul style="list-style-type: none"> Way, bits[31:32-A], the number of the way to operate on. Set, bits[B-1:L], the number of the set to operate on. Bits[L-1:4] are RES0 . $A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$. ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.	28 {x}
[3:1]	Level	Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.	xxx
[0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is **CONSTRAINED UNPREDICTABLE** and the implemented behavior is:

- The instruction performs cache maintenance on a single arbitrary cache line.

DC CSW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1010	0b010

Accessibility

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is **CONSTRAINED UNPREDICTABLE** and the implemented behavior is:

- The instruction performs cache maintenance on a single arbitrary cache line.

DC CSW, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TSW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_SetWay);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_SetWay);

```

A.2.4.12 DC CISW, Data or unified Cache line Clean and Invalidate by Set/Way

Clean and Invalidate data cache by set/way.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-210: AARCH64_DC_CISW bit assignments

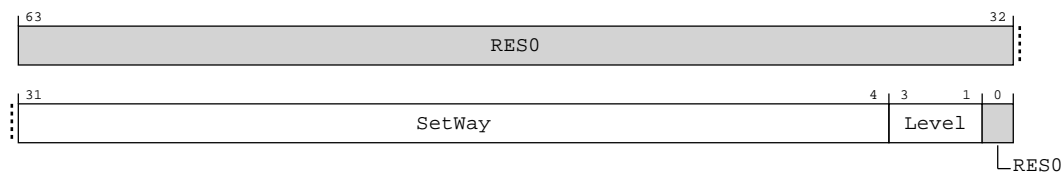


Table A-563: DC CISW bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:4]	SetWay	Contains two fields: <ul style="list-style-type: none">Way, bits[31:32-A], the number of the way to operate on.Set, bits[B-1:L], the number of the set to operate on. Bits[L-1:4] are RES0 . $A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$. ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.	28 {x}
[3:1]	Level	Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.	xxx
[0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is **CONSTRAINED UNPREDICTABLE** and the implemented behavior is:

- The instruction performs cache maintenance on a single arbitrary cache line.

DC C1SW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1110	0b010

Accessibility

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is **CONSTRAINED UNPREDICTABLE** and the implemented behavior is:

- The instruction performs cache maintenance on a single arbitrary cache line.

DC C1SW, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TSW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate,
        CacheOpScope_SetWay);
    elseif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate,
        CacheOpScope_SetWay);
```

A.2.4.13 TLBI VMALE1OS, TLB Invalidate by VMID, All at stage 1, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.



When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.



Note

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBI_AllAttr);
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
    Shareability_OSH, TLBI_AllAttr);

```

A.2.4.14 TLBI VAE1OS, TLB Invalidate by VA, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When



Note

a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-211: AARCH64_TLBI_VAE1OS bit assignments

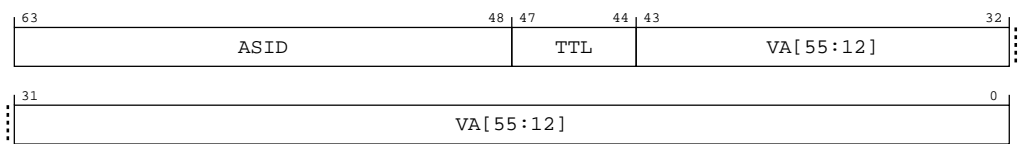


Table A-566: TLBI VAE1OS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b001

Accessibility

TLBI VAE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_OSH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.15 TLBI ASIDE1OS, TLB Invalidate by ASID, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.

The entry would be used with the current VMID and would be required to translate an address using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-212: AARCH64_TLBI_ASIDE1OS bit assignments

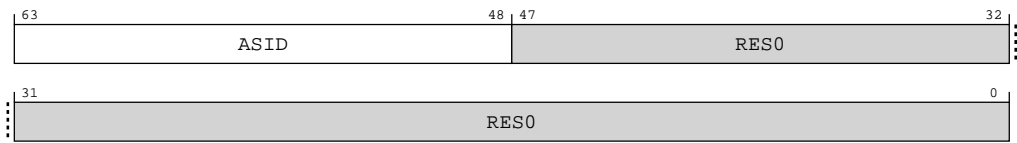


Table A-568: TLBI ASIDE1OS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this System instruction. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16{x}
[47:0]	RES0	Reserved	RES0

Access

TLBI ASIDE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b010

Accessibility

TLBI ASIDE1OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_OSH, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBI_AllAttr, X[t, 64]);
```

A.2.4.16 TLBI VAAE1OS, TLB Invalidate by VA, All ASID, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When



a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.



For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-213: AARCH64_TLBI_VAAE1OS bit assignments

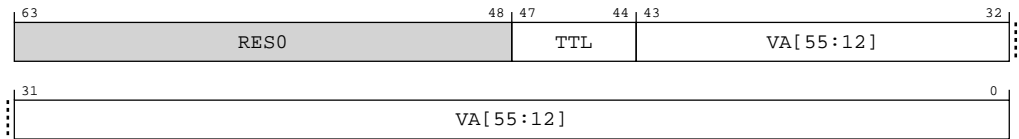


Table A-570: TLBI VAAE1OS bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b011

Accessibility

TLBI VAAE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.17 TLBI VALE1OS, TLB Invalidate by VA, Last level, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When

**Note**

a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-214: AARCH64_TLBI_VALE1OS bit assignments

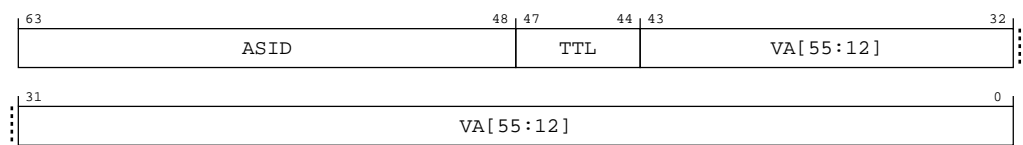


Table A-572: TLBI VALE1OS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b101

Accessibility

TLBI VALE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_OSH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.18 TLBI VAALE1OS, TLB Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When

**Note**

a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.

**Note**

For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-215: AARCH64_TLBI_VAALE1OS bit assignments

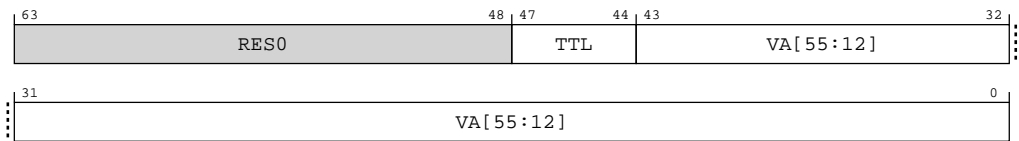


Table A-574: TLBI VAALE1OS bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0001	0b111

Accessibility

TLBI VAALE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.19 TLBI RVAE1IS, TLB Range Invalidate by VA, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[\text{BaseADDR} (5 \times \text{SCALE} + 1) \times \text{Translation_Granule_Size}]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 00000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-216: AARCH64_TLBI_RVAE1IS bit assignments

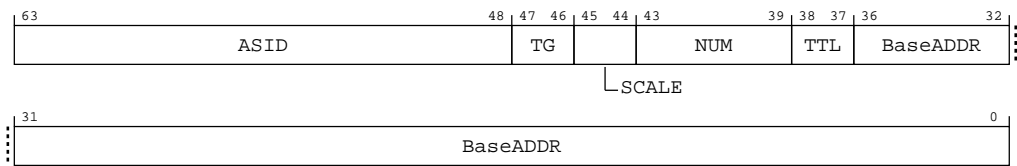


Table A-576: TLBI RVAE1IS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	<p>ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.</p> <p>Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.</p> <p>If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.</p>	16 {x}
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b001

Accessibility

TLBI RVAE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_ISH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.20 TLBI RVAE1IS, TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-217: AARCH64_TLBI_RVAAE1IS bit assignments

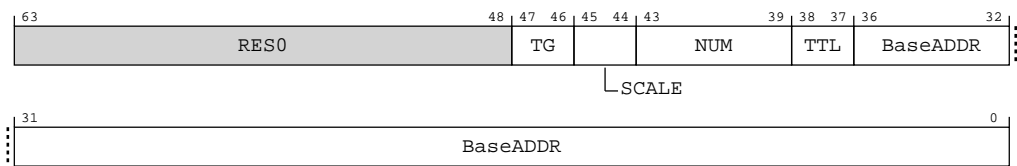


Table A-578: TLBI RVAAE1IS bit descriptions

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

Bits	Name	Description	Reset
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVAAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b011

Accessibility

TLBI RVAAE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.21 TLBI RVAE1IS, TLB Range Invalidate by VA, Last level, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $TTL=01$ and $BaseADDR[29:12]$ is not equal to 00000000000000000000 .

- If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-218: AARCH64_TLBI_RVALE1IS bit assignments

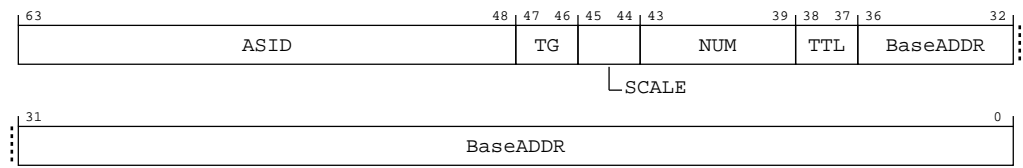


Table A-580: TLBI RVALE1IS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16{x}

Bits	Name	Description	Reset
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b101

Accessibility

TLBI RVALE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.22 TLBI RVALE1IS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- The entry is within the address range determined by the formula $[\text{BaseADDR} (5 \times \text{SCALE} + 1) * \text{Translation_Granule_Size}]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.

- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-219: AARCH64_TLBI_RVAALE1IS bit assignments

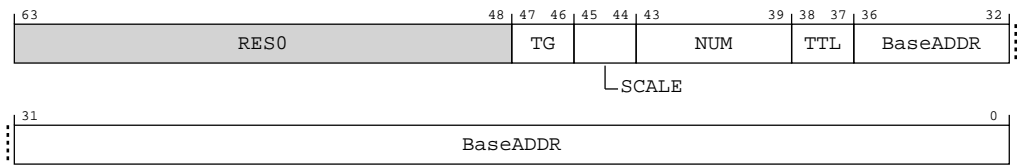


Table A-582: TLBI RVAALE1IS bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVAALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0010	0b111

Accessibility

TLBI RVAALE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_ISH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.23 TLBI VMALLE1IS, TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

From Armv8.4, when



Note

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.



Note

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is CONSTRAINED UNPREDICTABLE, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBI_AllAttr);
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
    Shareability_ISH, TLBI_AllAttr);

```

A.2.4.24 TLBI VAE1IS, TLB Invalidate by VA, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

From Armv8.4, when



Note

a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-220: AARCH64_TLBI_VAE1IS bit assignments

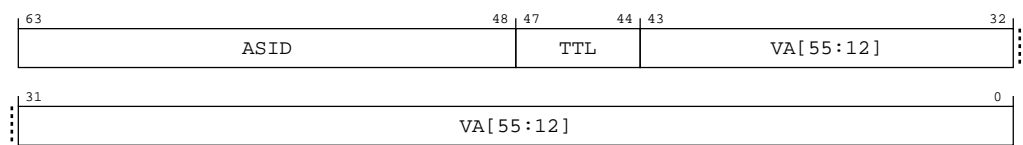


Table A-585: TLBI VAE1IS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16{x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b001

Accessibility

TLBI VAE1IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_ISH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
```

A.2.4.25 TLBI ASIDE1IS, TLB Invalidate by ASID, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.

The entry would be used with the current VMID and would be required to translate an address using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-221: AARCH64_TLBI_ASIDE1IS bit assignments

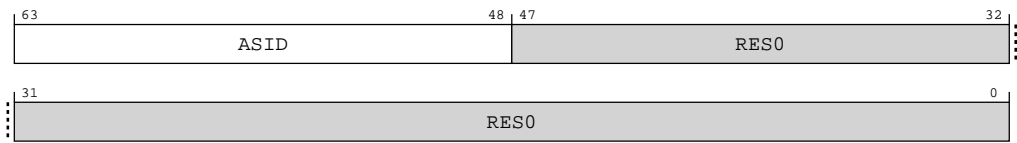


Table A-587: TLBI ASIDE1IS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this System instruction. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16{x}
[47:0]	RES0	Reserved	RES0

Access

TLBI ASIDE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b010

Accessibility

TLBI ASIDE1IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_ISH, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBI_AllAttr, X[t, 64]);
```

A.2.4.26 TLBI VAAE1IS, TLB Invalidate by VA, All ASID, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

From Armv8.4, when



a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.



For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-222: AARCH64_TLBI_VAAE1IS bit assignments

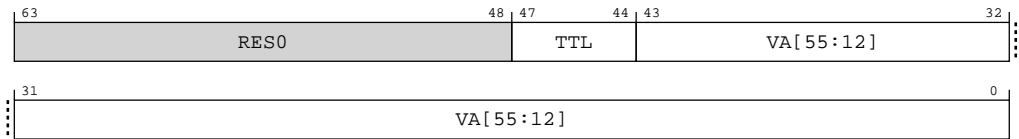


Table A-589: TLBI VAAE1IS bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAAE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b011

Accessibility

TLBI VAAE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.27 TLBI VALE1IS, TLB Invalidate by VA, Last level, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

From Armv8.4, when

**Note**

a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.
-

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-223: AARCH64_TLBI_VALE1IS bit assignments

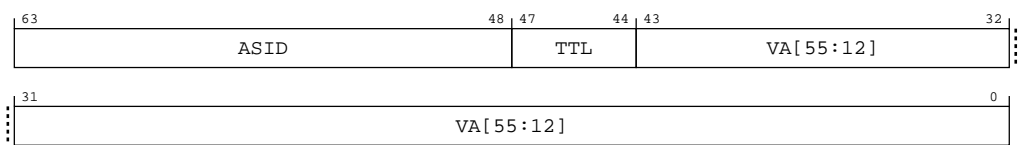


Table A-591: TLBI VALE1IS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b101

Accessibility

TLBI VALE1IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_ISH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
            TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
```

A.2.4.28 TLBI VAALE1IS, TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable


Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.


From Armv8.4, when



Note

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.



Note

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-224: AARCH64_TLBI_VAALE1IS bit assignments

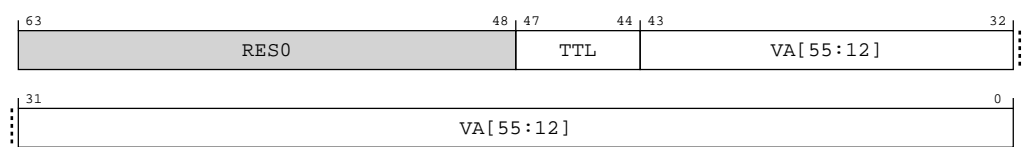


Table A-593: TLBI VAALE1IS bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAALE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b111

Accessibility

TLBI VAALE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.29 TLBI RVAE1OS, TLB Range Invalidate by VA, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[\text{BaseADDR} (5 \times \text{SCALE} + 1) \times \text{Translation_Granule_Size}]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-225: AARCH64_TLBI_RVAE1OS bit assignments

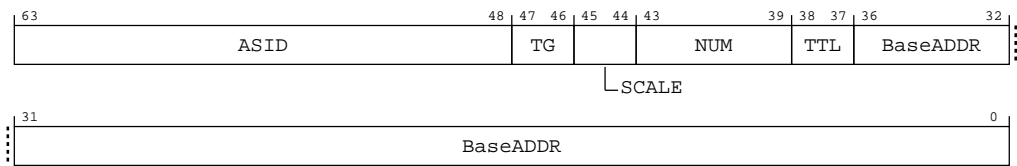


Table A-595: TLBI RVAE1OS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	<p>ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.</p> <p>Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.</p> <p>If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.</p>	16 {x}
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b001

Accessibility

TLBI RVAE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_OSH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elseif PSTATE.EL == EL2 then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.30 TLBI RVAE1OS, TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-226: AARCH64_TLBI_RVAAE1OS bit assignments

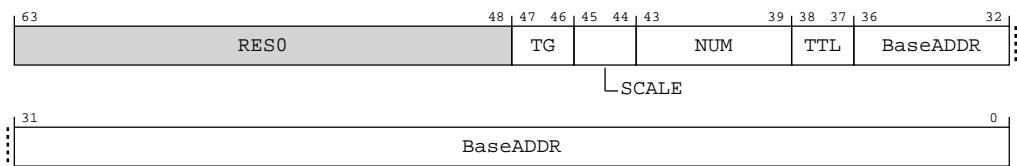


Table A-597: TLBI RVAAE1OS bit descriptions

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

Bits	Name	Description	Reset
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVAAE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b011

Accessibility

TLBI RVAAE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_OSH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.31 TLBI RVAE1OS, TLB Range Invalidate by VA, Last level, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR (5 \cdot SCALE + 1) \cdot Translation_Granule_Size]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $TTL=01$ and $BaseADDR[29:12]$ is not equal to 00000000000000000000 .

- If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-227: AARCH64_TLBI_RVALE1OS bit assignments

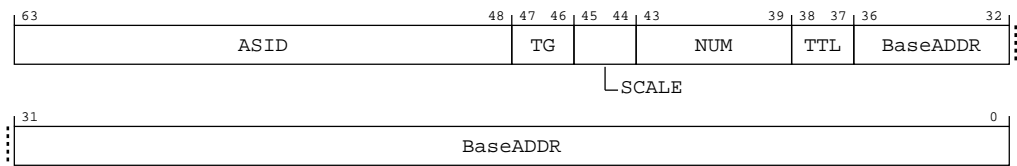


Table A-599: TLBI RVALE1OS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16{x}

Bits	Name	Description	Reset
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b101

Accessibility

TLBI RVALE1OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_OSH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
```

A.2.4.32 TLBI RVALE1OS, TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- The entry is within the address range determined by the formula $[\text{BaseADDR} (5 \times \text{SCALE} + 1) * \text{Translation_Granule_Size}]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

When

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.

- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-228: AARCH64_TLBI_RVAALE1OS bit assignments

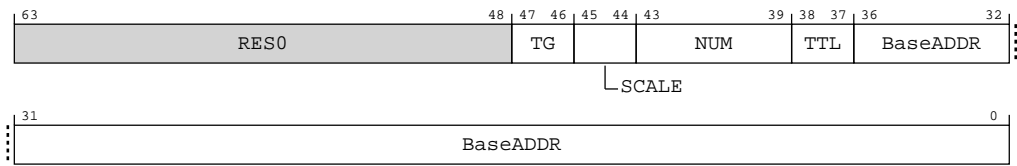


Table A-601: TLBI RVAALE1OS bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVAALE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0101	0b111

Accessibility

TLBI RVAALE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
            Shareability_OSH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_OSH,
            TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.33 TLBI RVAE1, TLB Range Invalidate by VA, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[\text{BaseADDR} (5 * \text{SCALE} + 1) * \text{Translation_Granule_Size}]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $\text{TTL} == 01$ and $\text{BaseADDR}[29:12]$ is not equal to 000000000000000000.
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[20:12]$ is not equal to 000000000.
- For the 16K translation granule:
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $\text{TTL} == 01$ and $\text{BaseADDR}[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[28:16]$ is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-229: AARCH64_TLBI_RVAE1 bit assignments

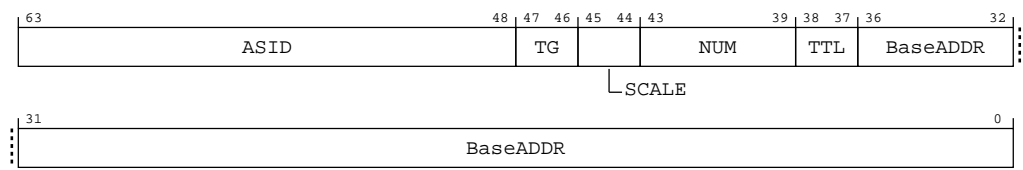


Table A-603: TLBI RVAE1 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b001

Accessibility

TLBI RVAE1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FB == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    endif
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
endif

```


A.2.4.34 TLBI RVAAE1, TLB Range Invalidate by VA, All ASID, EL1

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $TTL == 01$ and $BaseADDR[29:12]$ is not equal to 000000000000000000.
 - If $TTL == 10$ and $BaseADDR[20:12]$ is not equal to 0000000000.
- For the 16K translation granule:
 - If $TTL == 10$ and $BaseADDR[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $TTL == 01$ and $BaseADDR[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $TTL == 10$ and $BaseADDR[28:16]$ is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-230: AARCH64_TLBI_RVAAE1 bit assignments

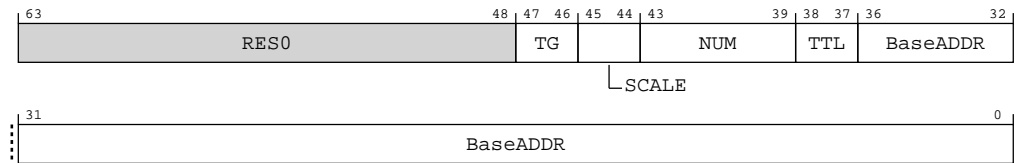


Table A-605: TLBI RVAAE1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx

Bits	Name	Description	Reset
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVAAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b011

Accessibility

TLBI RVAAE1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FB == '1' then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.35 TLBI RVALE1, TLB Range Invalidate by VA, Last level, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula $[BaseADDR (5 \times SCALE + 1) \times Translation_Granule_Size]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-231: AARCH64_TLBI_RVALE1 bit assignments

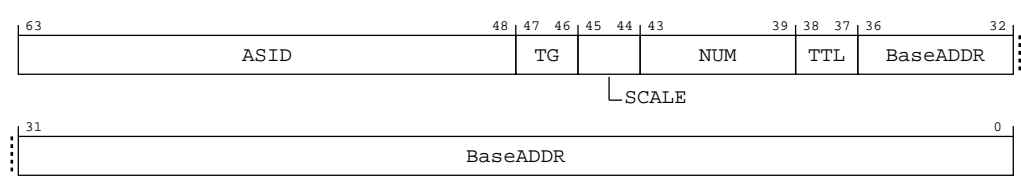


Table A-607: TLBI RVALE1 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	<p>ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.</p> <p>Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.</p> <p>If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.</p>	16 {x}
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b101

Accessibility

TLBI RVALE1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FB == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.36 TLBI RVALE1, TLB Range Invalidate by VA, All ASID, Last level, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- The entry is within the address range determined by the formula $[\text{BaseADDR} (5 * \text{SCALE} + 1) * \text{Translation_Granule_Size}]$.

The entry would be used with the current VMID and would be required to translate any of the VAs in the specified address range using the EL1&0 translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.

- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-232: AARCH64_TLBI_RVAALE1 bit assignments

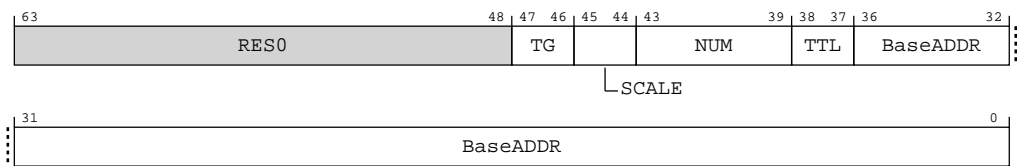


Table A-609: TLBI RVAALE1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVAALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b111

Accessibility

TLBI RVAALE1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FB == '1' then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
    endif
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_RVAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
endif

```


A.2.4.37 TLBI VMALLE1, TLB Invalidate by VMID, All at stage 1, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.



For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLE1{, <Xt>}

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

```

if HCR_EL2.TTLB == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif HCR_EL2.FB == '1' then
    AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
Shareability_ISH, TLBI_AllAttr);
else
    AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
Shareability_NSH, TLBI_AllAttr);
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_VMALL(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
Shareability_NSH, TLBI_AllAttr);

```

A.2.4.38 TLBI VAE1, TLB Invalidate by VA, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-233: AARCH64_TLBI_VAE1 bit assignments

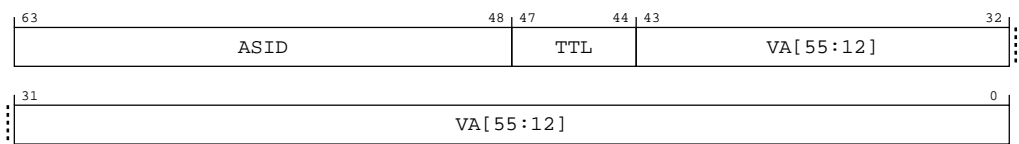


Table A-612: TLBI VAE1 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b001

Accessibility

TLBI VAE1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FB == '1' then
        AArch64.TLBI VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL2 then
    AArch64.TLBI VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.39 TLBI ASIDE1, TLB Invalidate by ASID, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.

The entry would be used with the current VMID and would be required to translate an address using the EL1&0 translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-234: AARCH64_TLBI_ASIDE1 bit assignments

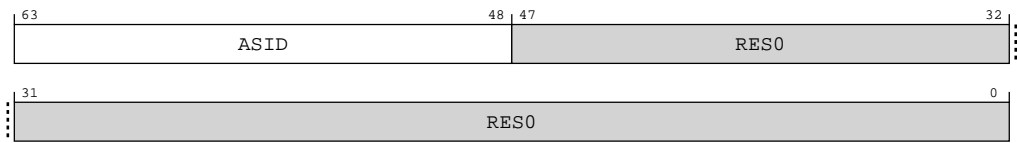


Table A-614: TLBI ASIDE1 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this System instruction. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16{x}
[47:0]	RES0	Reserved	RES0

Access

TLBI ASIDE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b010

Accessibility

TLBI ASIDE1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FB == '1' then
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_ASID(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBI_AllAttr, X[t, 64]);
```

A.2.4.40 TLBI VAAE1, TLB Invalidate by VA, All ASID, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.



For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-235: AARCH64_TLBI_VAAE1 bit assignments

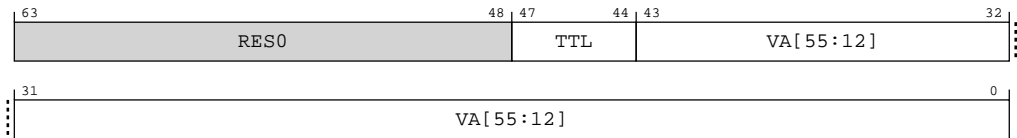


Table A-616: TLBI VAAE1 bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAAE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b011

Accessibility

TLBI VAAE1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FB == '1' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
```

A.2.4.41 TLBI VALE1, TLB Invalidate by VA, Last level, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-236: AARCH64_TLBI_VALE1 bit assignments

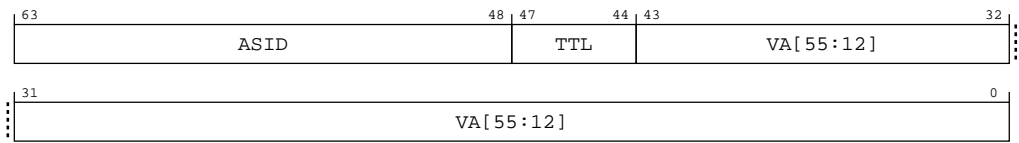


Table A-618: TLBI VALE1 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b101

Accessibility

TLBI VALE1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FB == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
    TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
```


A.2.4.42 TLBI VAALE1, TLB Invalidate by VA, All ASID, Last level, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.

The entry would be used with the current VMID and would be required to translate the specified VA using the EL1&O translation regime for the Security state.

The invalidation applies to the PE that executes this System instruction.



Note

For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group
System instructions

Access type
See bit descriptions

Bit descriptions

Figure A-237: AARCH64_TLBI_VAALE1 bit assignments

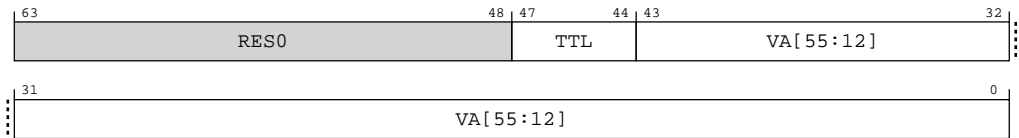


Table A-620: TLBI VAALE1 bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b111

Accessibility

TLBI VAALE1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FB == '1' then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        AArch64.TLBI_VAA(SecurityStateAtEL(EL1), Regime_EL10, VMID[], Shareability_NSH,
        TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
```

A.2.4.43 SYS IMP_BPI, Branch Predictor Invalidation Operation

Invalidates the state of all branch predictors.

Executing this instruction causes all state of the dynamic branch prediction structures (TAGE conditional predictor, CRS, BTAC, nano predictors) to be invalidated, regardless of their context. The invalidation operation takes 128 cycles to execute. In the meantime, any branches encountered on instruction fetches are predicted using static prediction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-238: AARCH64_SYS_IMP_BPI bit assignments

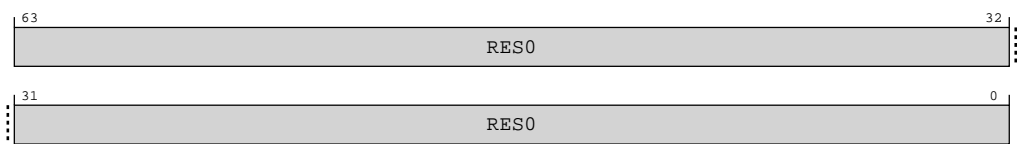


Table A-622: SYS_IMP_BPI bit descriptions

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

Access

SYS #0, C15, C1, #1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1111	0b0001	0b001

Accessibility

SYS #0, C15, C1, #1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.BPRED == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_BPI(X[t, 64]);
elseif PSTATE.EL == EL2 then
    SYS_IMP_BPI(X[t, 64]);
```

A.2.4.44 SYS_IMP_CDBGDCT, L1 Data Cache Tag Read Operation

Read contents of the L1 Data Cache Tag Memory.

The 64 bits of cache data returns in AArch64-IMP_CDBGDR0_EL1 and AArch64-IMP_CDBGDR1_EL1 is zeroed.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group
System instructions

Access type
See bit descriptions

Bit descriptions

Figure A-239: AARCH64_SYS_IMP_CDBGDCT bit assignments

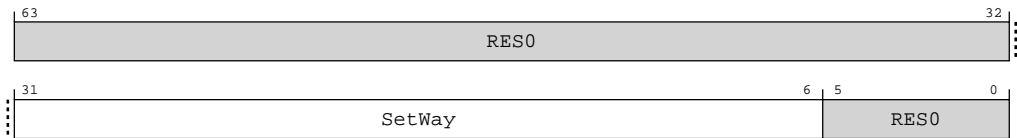


Table A-624: SYS_IMP_CDBGDCT bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	SetWay	Contains two fields: <ul style="list-style-type: none">Way, bits[31:30], the number of the way to operate on.Set, bits[S-1:6], the number of the set to operate on. Bits[29:S] are RES0. The Set width depends on the implemented cache size. The unused bits are RES0. The index is specified as below: <ul style="list-style-type: none">S=12 for a 16KB cache.S=13 for a 32KB cache.S=14 for a 64KB cache.	26 {x}
[5:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C2, #0{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0010	0b000

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C2, #0{, <Xt>}

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

```
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CDBGDCT(X[t, 64]);
elseif PSTATE.EL == EL2 then
    SYS_IMP_CDBGDCT(X[t, 64]);
```

A.2.4.45 SYS_IMP_CDBGICT, L1 Instruction Cache Tag Read Operation

Read contents of the L1 Instruction Cache Tag Memory.

The 64 bits of cache data returns in AArch64-IMP_CDBGDR0_EL1 and AArch64-IMP_CDBGDR1_EL1 is zeroed.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-240: AARCH64_SYS_IMP_CDBGICT bit assignments

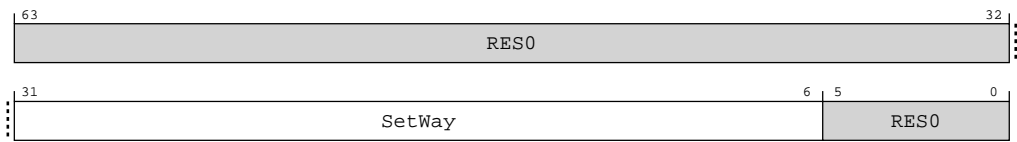


Table A-626: SYS_IMP_CDBGICT bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:6]	SetWay	<p>Contains two fields:</p> <ul style="list-style-type: none"> Way, bits[31:30], the number of the way to operate on. Set, bits[S-1:6], the number of the set to operate on. <p>Bits[29:S] are RES0.</p> <p>The Set width depends on the implemented cache size. The unused bits are RES0. The index is specified as below:</p> <ul style="list-style-type: none"> S=12 for a 16KB cache. S=13 for a 32KB cache. S=14 for a 64KB cache. 	26{x}
[5:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C2, #1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0010	0b001

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C2, #1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CDBGICT(X[t, 64]);
elseif PSTATE.EL == EL2 then
    SYS_IMP_CDBGICT(X[t, 64]);

```

A.2.4.46 SYS_IMP_CDBGTT, TLB Tag Read Operation

Read contents of the L2 TLB Tag Memory and Walk cache RAM Tag Memory.

The 64 bits of cache data returns in AArch64-IMP_CDBGDR0_EL1 and AArch64-IMP_CDBGDR1_EL1 is zeroed.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-241: AARCH64_SYS_IMP_CDBGTT bit assignments

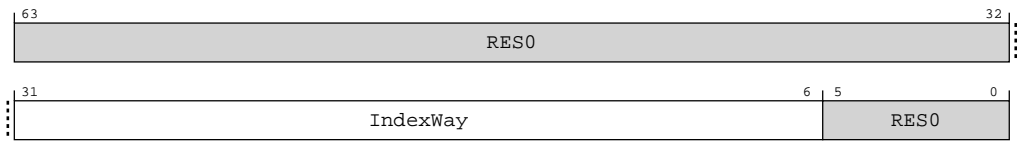


Table A-628: SYS_IMP_CDBGTT bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	IndexWay	Contains two fields: <ul style="list-style-type: none">Way, bits[31:30], the number of the way to operate on.Index, bits[14:6], the number of the index to operate on. Bits[29:15] are RES0. The index field is used to select the index from the TLB, or walk cache. <ul style="list-style-type: none">When index is in range 0x000 - 0xFF, Main TLB is selected.When index is in range 0x100 - 0x107, walk cache is selected.	26{x}
[5:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C2, #2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0010	0b010

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C2, #2{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CDBGTT(X[t, 64]);
elsif PSTATE.EL == EL2 then
    SYS_IMP_CDBGTT(X[t, 64]);
```

A.2.4.47 SYS_IMP_CLUSTERCDBGL2T, L2 Cache Tag Read Operation

Read contents of the L2 Cache Tag Memory.

The 64 bits of cache data returns in AArch64-IMP_CLUSTERCDBGDRO_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-242: AARCH64_SYS_IMP_CLUSTERCDBGL2T bit assignments

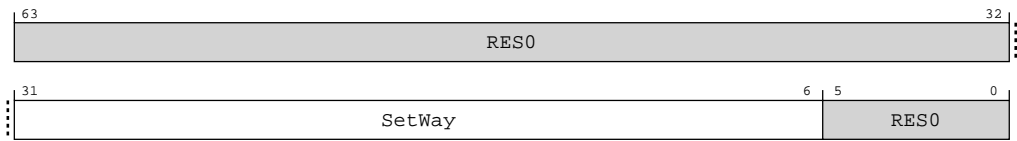


Table A-630: SYS_IMP_CLUSTERCDBGL2T bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:6]	SetWay	Contains two fields: <ul style="list-style-type: none"> Way, bits[31:29], the number of the way to operate on. Set, bits[18:6], the number of the set to operate on. Bits[28:19] are RES0 . Note: The set index width depends on the implemented cache size. The unused bits are RES0 .	26 {x}
[5:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.



Note

If this instruction is executed when the L2 cache size is configured 0, the value returned is 0.

SYS #2, C15, C2, #3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0010	0b011

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.



Note

If this instruction is executed when the L2 cache size is configured 0, the value returned is 0.

SYS #2, C15, C2, #3{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CLUSTERCDBGL2T(X[t, 64]);
    elsif PSTATE.EL == EL2 then
        SYS_IMP_CLUSTERCDBGL2T(X[t, 64]);

```

A.2.4.48 SYS_IMP_CLUSTERCDBGL2DT, L2 Cache Duplicate L1 Tag Read Operation

Read contents of the L2 Cache Duplicate L1 Tag Memory.

The 64 bits of cache data returns in AArch64-IMP_CLUSTERCDBGDRO_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-243: AARCH64_SYS_IMP_CLUSTERCDBGL2DT bit assignments

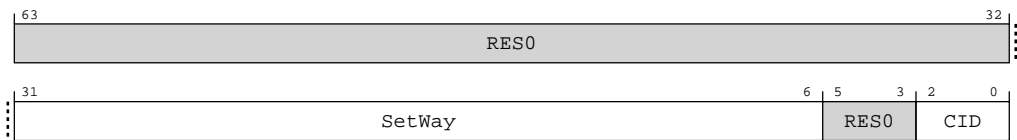


Table A-632: SYS_IMP_CLUSTERCDBGL2DT bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	SetWay	Contains two fields: <ul style="list-style-type: none">Way, bits[31:30], the number of the way to operate on.Set, bits[13:6], the number of the set to operate on. Bits[29:14] are RES0. Note: The set index width depends on the implemented cache size. The unused bits are RES0.	2 6 { x }
[5:3]	RES0	Reserved	RES0
[2:0]	CID	The core ID whose duplicate L1 tags are to be read.	xxx

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

If this instruction is executed with a core number that is larger than the value supported by the implementation, then the instruction performs a read from the duplicate tags of an arbitrary core.

SYS #2, C15, C3, #3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0011	0b011

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

If this instruction is executed with a core number that is larger than the value supported by the implementation, then the instruction performs a read from the duplicate tags of an arbitrary core.

SYS #2, C15, C3, #3{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CLUSTERCDBGL2DT(X[t, 64]);
elsif PSTATE.EL == EL2 then
        SYS_IMP_CLUSTERCDBGL2DT(X[t, 64]);

```

A.2.4.49 SYS_IMP_CLUSTERCDBGLCU DT, LCU Duplicate L1 Tag Read Operation

Read contents of the LCU Duplicate L1 Tag Memory.

The 64 bits of cache data returns in AArch64-IMP_CLUSTERCDBGDRO_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-244: AARCH64_SYS_IMP_CLUSTERCDBGLCUDT bit assignments

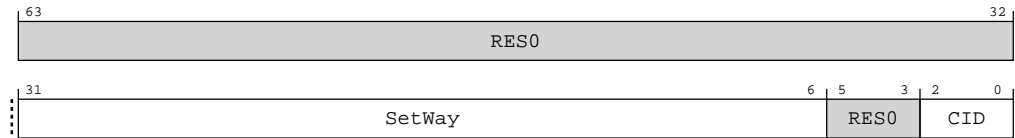


Table A-634: SYS_IMP_CLUSTERCDBGLCUDT bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	SetWay	Contains two fields: <ul style="list-style-type: none"> Way, bits[31:30], the number of the way to operate on. Set, bits[13:6], the number of the set to operate on. Bits[29:14] are RES0. Note: The set index width depends on the implemented cache size. The unused bits are RES0.	26 {x}
[5:3]	RES0	Reserved	RES0
[2:0]	CID	The core ID whose duplicate L1 tags are to be read.	xxx

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

If this instruction is executed with a core number that is larger than the value supported by the implementation, then the instruction performs a read from the duplicate tags of an arbitrary core.



Note

If this instruction is executed when the LLRAM is not implemented, the value returned is **UNKNOWN**.

SYS #2, C15, C3, #4{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0011	0b100

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

If this instruction is executed with a core number that is larger than the value supported by the implementation, then the instruction performs a read from the duplicate tags of an arbitrary core.



If this instruction is executed when the LLRAM is not implemented, the value returned is UNKNOWN.

SYS #2, C15, C3, #4{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CLUSTERCDBGLCUdT(X[t, 64]);
elsif PSTATE.EL == EL2 then
    SYS_IMP_CLUSTERCDBGLCUdT(X[t, 64]);
```

A.2.4.50 SYS_IMP_CDBGDCD, L1 Data Cache Data Read Operation

Read contents of the L1 Data Cache Data Memory.

The 128 bits of cache data returns in AArch64-IMP_CDBGDR0_EL1 and AArch64-IMP_CDBGDR1_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-245: AARCH64_SYS_IMP_CDBGDCD bit assignments

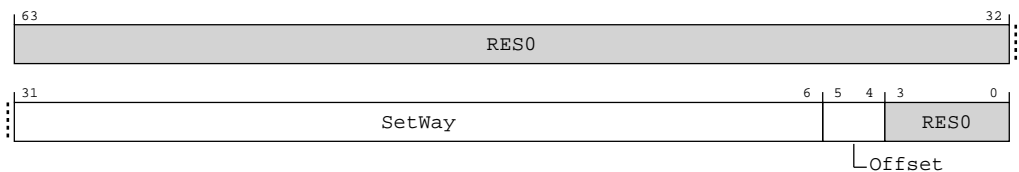


Table A-636: SYS IMP_CDBGDCD bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	SetWay	<p>Contains two fields:</p> <ul style="list-style-type: none"> Way, bits[31:30], the number of the way to operate on. Set, bits[S-1:6], the number of the set to operate on. <p>Bits[29:S] are RES0.</p> <p>The Set width depends on the implemented cache size. The unused bits are RES0. The index is specified as below:</p> <ul style="list-style-type: none"> S=12 for a 16KB cache. S=13 for a 32KB cache. S=14 for a 64KB cache. 	26{x}
[5:4]	Offset	Cache data element offset	xx
[3:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C4, #0{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0100	0b000

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C4, #0{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CDBGDCD(X[t, 64]);
elseif PSTATE.EL == EL2 then
    SYS_IMP_CDBGDCD(X[t, 64]);

```

A.2.4.51 SYS IMP_CDBGICD, L1 Instruction Cache Data Read Operation

Read contents of the L1 Instruction Cache Data Memory.

The 128 bits of cache data returns in AArch64-IMP_CDBGDR0_EL1 and AArch64-IMP_CDBGDR1_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-246: AARCH64_SYS_IMP_CDBGICD bit assignments

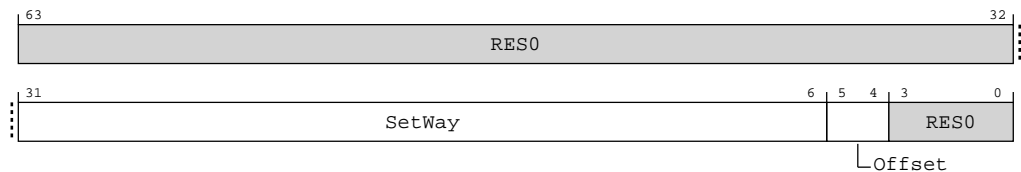


Table A-638: SYS_IMP_CDBGICD bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	SetWay	Contains two fields: <ul style="list-style-type: none">Way, bits[31:30], the number of the way to operate on.Set, bits[S-1:6], the number of the set to operate on. Bits[29:S] are RES0. The Set width depends on the implemented cache size. The unused bits are RES0. The index is specified as below: <ul style="list-style-type: none">S=12 for a 16KB cache.S=13 for a 32KB cache.S=14 for a 64KB cache.	26{x}
[5:4]	Offset	Cache data element offset	xx
[3:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C4, #1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0100	0b001

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.
SYS #2, C15, C4, #1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CDBGICD(X[t, 64]);
elseif PSTATE.EL == EL2 then
    SYS_IMP_CDBGICD(X[t, 64]);
```

A.2.4.52 SYS_IMP_CDBGTD, TLB Data Read Operation

Read contents of the TLB Data Memory.

The 64 bits of cache data returns in AArch64-IMP_CDBGDR0_EL1 and AArch64-IMP_CDBGDR1_EL1 is zeroed.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-247: AARCH64_SYS_IMP_CDBGTD bit assignments

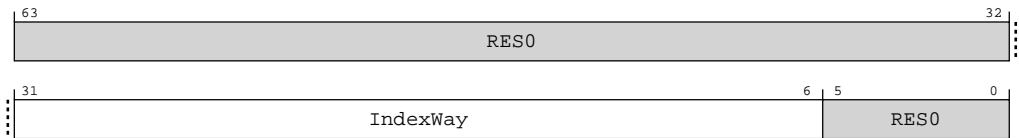


Table A-640: SYS_IMP_CDBGTD bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	IndexWay	<p>Contains two fields:</p> <ul style="list-style-type: none"> Way, bits[31:30], the number of the way to operate on. Index, bits[14:6], the number of the index to operate on. <p>Bits[29:15] are RES0.</p> <p>The index field is used to select the index from the TLB, or walk cache.</p> <ul style="list-style-type: none"> When index is in range 0x000 - 0x0FF, Main TLB is selected. When index is in range 0x100 - 0x107, walk cache is selected. 	26{x}
[5:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C4, #2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0100	0b010

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.

SYS #2, C15, C4, #2{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CDBGTD(X[t, 64]);
elseif PSTATE.EL == EL2 then
    SYS_IMP_CDBGTD(X[t, 64]);

```

A.2.4.53 SYS_IMP_CLUSTERCDBGL2D, L2 Cache Data Read Operation

Read contents of the L2 Cache Data Memory.

The 64 bits of cache data returns in AArch64-IMP_CLUSTERCDBGDR0_EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-248: AARCH64_SYS_IMP_CLUSTERCDBGL2D bit assignments

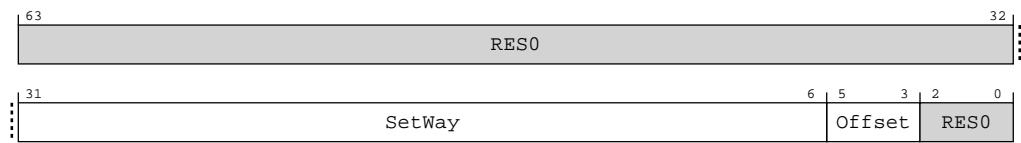



Table A-642: SYS_IMP_CLUSTERCDBGL2D bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:6]	SetWay	Contains two fields: <ul style="list-style-type: none">Way, bits[31:29], the number of the way to operate on.Set, bits[18:6], the number of the set to operate on. Bits[28:19] are RES0. Note: The set index width depends on the implemented cache size. The unused bits are RES0.	2 6 { x }
[5:3]	Offset	Cache data element offset	xxx
[2:0]	RES0	Reserved	RES0

Access

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.



Note

If this instruction is executed when the L2 cache size is configured 0, the value returned is 0.

SYS #2, C15, C4, #3{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b010	0b1111	0b0100	0b011

Accessibility

If this instruction is executed with a set or way argument that is larger than the value supported by the implementation, then the instruction performs a read on a single arbitrary cache line.



If this instruction is executed when the L2 cache size is configured 0, the value returned is 0.

SYS #2, C15, C4, #3{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TIDCP == '1' || ACTLR_EL2.CDBG == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        SYS_IMP_CLUSTERCDBG_L2D(X[t, 64]);
elseif PSTATE.EL == EL2 then
    SYS_IMP_CLUSTERCDBG_L2D(X[t, 64]);
```

A.2.4.54 CFP RCTX, Control Flow Prediction Restriction by Context

Control Flow Prediction Restriction by Context applies to all Control Flow Prediction Resources that predict execution based on information gathered within the target execution context or contexts.

Control flow predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.



This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configurations

This register is available in all configurations.

Attributes**Width**

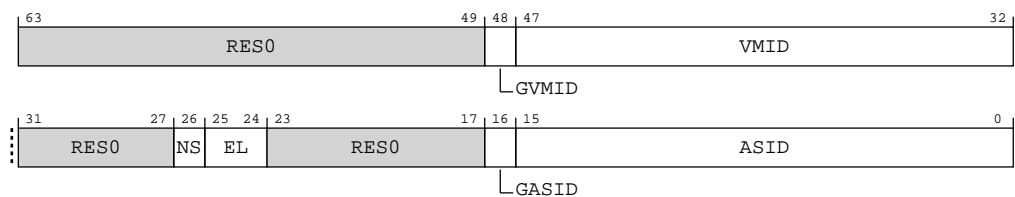
64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions**Figure A-249: AARCH64_CFP_RCTX bit assignments****Table A-644: CFP RCTX bit descriptions**

Bits	Name	Description	Reset
[63:49]	RES0	Reserved	RES0
[48]	GVMID	<p>Execution of this instruction applies to all VMIDs or a specified VMID.</p> <p>0b0 Applies to specified VMID for an ELO or EL1 target execution context.</p> <p>0b1 Applies to all VMIDs for an ELO or EL1 target execution context.</p> <p>For target execution contexts other than ELO or EL1, this field is RES0.</p> <p>If the instruction is executed at ELO or EL1, this field has an Effective value of 0.</p>	x
[47:32]	VMID	<p>Only applies when bit[48] is 0 and the target execution context is either:</p> <ul style="list-style-type: none"> EL1. ELO. <p>Otherwise this field is RES0.</p> <p>When the instruction is executed at EL1, this field is treated as the current VMID.</p> <p>When the instruction is executed at ELO, this field is treated as the current VMID.</p>	16 {x}
[31:27]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[26]	NS	Security State. Defined values are: 0b0 Secure state. 0b1 Non-secure state.	x
[25:24]	EL	Exception Level. Indicates the Exception level of the target execution context. 0b00 EL0. 0b01 EL1. 0b10 EL2. 0b11 EL3. If the instruction is executed at an Exception level lower than the specified level, or is specified to apply to a combination of Exception level and Security state that is not implemented, this instruction is treated as a NOP .	xx
[23:17]	RES0	Reserved	RES0
[16]	GASID	Execution of this instruction applies to all ASIDs or a specified ASID. 0b0 Applies to specified ASID for an EL0 target execution context. 0b1 Applies to all ASIDs for an EL0 target execution context. For target execution contexts other than EL0, this field is RES0 . If the instruction is executed at EL0, this field has an Effective value of 0.	x
[15:0]	ASID	Only applies for an EL0 target execution context and when bit[16] is 0. Otherwise, this field is RES0 . When the instruction is executed at EL0, this field is treated as the current ASID. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.	16{x}

Access

CFP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b100

Accessibility

CFP RCTX, <Xt>

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

```

if SCTL_EL1.EnRCTX == '0' then
  if HCR_EL2.TGE == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    AArch64.SystemAccessTrap(EL1, 0x18);
  else
    AArch64.RestrictPrediction(X[t, 64], RestrictType_ControlFlow);
elseif PSTATE.EL == EL1 then
  AArch64.RestrictPrediction(X[t, 64], RestrictType_ControlFlow);
elseif PSTATE.EL == EL2 then
  AArch64.RestrictPrediction(X[t, 64], RestrictType_ControlFlow);

```

A.2.4.55 DVP RCTX, Data Value Prediction Restriction by Context

Data Value Prediction Restriction by Context applies to all Data Value Prediction Resources that predict execution based on information gathered within the target execution context or contexts.



The prediction of the PSTATE.{N,Z,C,V} values is not considered a data value for this purpose.

Data value predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control speculative execution occurring after the instruction is complete and synchronized. This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.



This instruction does not require the invalidation of prediction structures so long as the behavior described for completion of this instruction is met by the implementation. On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-250: AARCH64_DVP_RCTX bit assignments

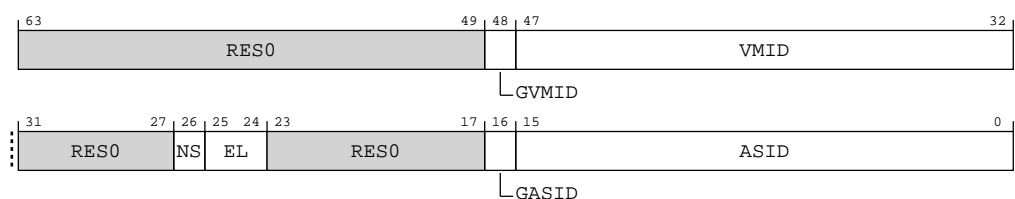


Table A-646: DVP RCTX bit descriptions

Bits	Name	Description	Reset
[63:49]	RES0	Reserved	RES0
[48]	GVMID	<p>Execution of this instruction applies to all VMIDs or a specified VMID.</p> <p>0b0 Applies to specified VMID for an ELO or EL1 target execution context.</p> <p>0b1 Applies to all VMIDs for an ELO or EL1 target execution context.</p> <p>For target execution contexts other than ELO or EL1, this field is RES0.</p> <p>If the instruction is executed at ELO or EL1, then this field has an Effective value of 0.</p>	x
[47:32]	VMID	<p>Only applies when bit[48] is 0 and the target execution context is either:</p> <ul style="list-style-type: none"> EL1. ELO. <p>Otherwise this field is RES0.</p> <p>When the instruction is executed at EL1, this field is treated as the current VMID.</p> <p>When the instruction is executed at ELO, this field is treated as the current VMID.</p>	16{x}
[31:27]	RES0	Reserved	RES0
[26]	NS	<p>Security State. Defined values are:</p> <p>0b0 Secure state.</p> <p>0b1 Non-secure state.</p>	x

Bits	Name	Description	Reset
[25:24]	EL	Exception Level. Indicates the Exception level of the target execution context. 0b00 EL0. 0b01 EL1. 0b10 EL2. 0b11 EL3. If the instruction is executed at an Exception level lower than the specified level, or is specified to apply to a combination of Exception level and Security state that is not implemented, this instruction is treated as a NOP .	xx
[23:17]	RES0	Reserved	RES0
[16]	GASID	Execution of this instruction applies to all ASIDs or a specified ASID. 0b0 Applies to specified ASID for an EL0 target execution context. 0b1 Applies to all ASIDs for an EL0 target execution context. For target execution contexts other than EL0, this field is RES0 . If the instruction is executed at EL0, this field has an Effective value of 0.	x
[15:0]	ASID	Only applies for an EL0 target execution context and when bit[16] is 0. Otherwise this field is RES0 . When the instruction is executed at EL0, this field is treated as the current ASID. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.	16{x}

Access

DVP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b101

Accessibility

DVP RCTX, <Xt>

```

if PSTATE.EL == EL0 then
    if SCTL_EL1.EnRCTX == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_DataValue);
    elseif PSTATE.EL == EL1 then

```

```
AArch64.RestrictPrediction(X[t, 64], RestrictType_DataValue);
elseif PSTATE.EL == EL2 then
    AArch64.RestrictPrediction(X[t, 64], RestrictType_DataValue);
```

A.2.4.56 CPP RCTX, Cache Prefetch Prediction Restriction by Context

Cache Prefetch Prediction Restriction by Context applies to all Cache Allocation Resources that predict cache allocations based on information gathered within the target execution context or contexts.

The actions of code in the target execution context or contexts appearing in program order before the instruction cannot exploitatively control cache prefetch predictions occurring after the instruction is complete and synchronized.

This instruction applies to all:

- Instruction caches.
- Data caches.
- TLB prefetching hardware used by the executing PE that applies to the supplied context or contexts.

This instruction is guaranteed to be complete following a DSB that covers both read and write behavior on the same PE as executed the original restriction instruction, and a subsequent context synchronization event is required to ensure that the effect of the completion of the instructions is synchronized to the current execution.



Note

This instruction does not require the invalidation of Cache Allocation Resources so long as the behavior described for completion of this instruction is met by the implementation.

On some implementations the instruction is likely to take a significant number of cycles to execute. This instruction is expected to be used very rarely, such as on the roll-over of an ASID or VMID, but should not be used on every context switch.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-251: AARCH64_CPP_RCTX bit assignments

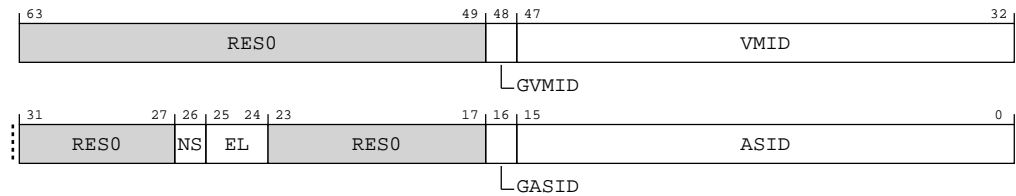


Table A-648: CPP RCTX bit descriptions

Bits	Name	Description	Reset
[63:49]	RES0	Reserved	RES0
[48]	GVMID	<p>Execution of this instruction applies to all VMIDs or a specified VMID.</p> <p>0b0 Applies to specified VMID for an ELO or EL1 target execution context.</p> <p>0b1 Applies to all VMIDs for an ELO or EL1 target execution context.</p> <p>For target execution contexts other than ELO and EL1, this field is RES0.</p> <p>If the instruction is executed at ELO or EL1, this field has an Effective value of 0.</p>	x
[47:32]	VMID	<p>Only applies when bit[48] is 0 and the target execution context is either:</p> <ul style="list-style-type: none"> EL1. ELO. <p>Otherwise this field is RES0.</p> <p>When the instruction is executed at EL1, this field is treated as the current VMID.</p> <p>When the instruction is executed at ELO, this field is treated as the current VMID.</p>	16{x}
[31:27]	RES0	Reserved	RES0
[26]	NS	<p>Security State. Defined values are:</p> <p>0b0 Secure state.</p> <p>0b1 Non-secure state.</p>	x

Bits	Name	Description	Reset
[25:24]	EL	Exception Level. Indicates the Exception level of the target execution context. 0b00 EL0. 0b01 EL1. 0b10 EL2. 0b11 EL3. If the instruction is executed at an Exception level lower than the specified level, or is specified to apply to a combination of Exception level and Security state that is not implemented, this instruction is treated as a NOP .	xx
[23:17]	RES0	Reserved	RES0
[16]	GASID	Execution of this instruction applies to all ASIDs or a specified ASID. 0b0 Applies to specified ASID for an EL0 target execution context. 0b1 Applies to all ASIDs for an EL0 target execution context. For target execution contexts other than EL0, this field is RES0 . If the instruction is executed at EL0, this field has an Effective value of 0.	x
[15:0]	ASID	Only applies for an EL0 target execution context and when bit[16] is 0. Otherwise, this field is RES0 . When the instruction is executed at EL0, this field is treated as the current ASID. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being affected only uses 8 bits.	16{x}

Access

CPP RCTX, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0011	0b111

Accessibility

CPP RCTX, <Xt>

```

if PSTATE.EL == EL0 then
    if SCTL_EL1.EnRCTX == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            AArch64.RestrictPrediction(X[t, 64], RestrictType_CachePrefetch);
    elseif PSTATE.EL == EL1 then

```



```
AArch64.RestrictPrediction(X[t, 64], RestrictType_CachePrefetch);
elseif PSTATE.EL == EL2 then
    AArch64.RestrictPrediction(X[t, 64], RestrictType_CachePrefetch);
```

A.2.4.57 DC ZVA, Data Cache Zero by VA

Zero data cache by address. Zeroes a naturally aligned block of N bytes, where the size of N is identified in AArch64-DCZID_ELO.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-252: AARCH64_DC_ZVA bit assignments

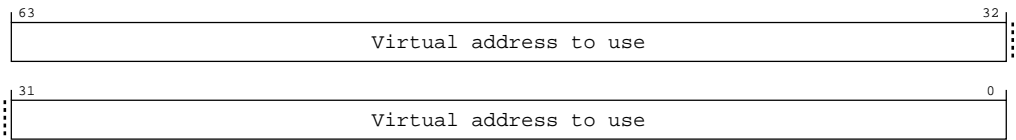


Table A-650: DC ZVA bit descriptions

Bits	Name	Description	Reset
[63:0]	VA	Virtual address to use. There is no alignment restriction on the address within the block of N bytes that is used.	64 { x }

Access

When this instruction is executed, it can generate memory faults or watchpoints which are prioritized in the same way as other memory-related faults or watchpoints. If a synchronous data abort fault or a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 0.

If the memory region being zeroed is any type of Device memory, this instruction generates an Alignment fault which is prioritized in the same way as other Alignment faults that are determined by the memory type.

This instruction applies to Normal memory regardless of cacheability attributes.

This instruction behaves as a set of Stores to each byte within the block being accessed, and so it:

- Generates a Permission fault if the translation system does not permit writes to the locations.
- Requires the same considerations for ordering and the management of coherency as any other store instructions.

DC ZVA, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0100	0b001

Accessibility

When this instruction is executed, it can generate memory faults or watchpoints which are prioritized in the same way as other memory-related faults or watchpoints. If a synchronous data abort fault or a watchpoint is generated, the CM bit in the ESR_ELx.ISS field is set to 0.

If the memory region being zeroed is any type of Device memory, this instruction generates an Alignment fault which is prioritized in the same way as other Alignment faults that are determined by the memory type.

This instruction applies to Normal memory regardless of cacheability attributes.

This instruction behaves as a set of Stores to each byte within the block being accessed, and so it:

- Generates a Permission fault if the translation system does not permit writes to the locations.
- Requires the same considerations for ordering and the management of coherency as any other store instructions.

DC ZVA, <Xt>

```

if PSTATE.EL == EL0 then
    if SCTLRL_EL1.DZE == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Data);
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TDZ == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.MemZero(X[t, 64], CacheType_Data);
    elsif PSTATE.EL == EL2 then
        AArch64.MemZero(X[t, 64], CacheType_Data);

```

A.2.4.58 IC IVAU, Instruction Cache line Invalidate by VA to PoU

Invalidate instruction cache by address to Point of Unification.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-253: AARCH64_IC_IVAU bit assignments

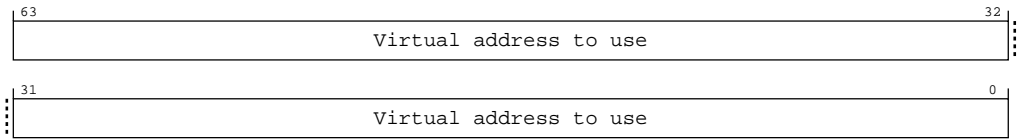


Table A-652: IC IVAU bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Virtual address to use. No alignment restrictions apply to this VA.	64 {x}

Access

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in *MMU faults generated by cache maintenance operations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see *The data cache maintenance instruction (DC)* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

IC IVAU{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0101	0b001

Accessibility

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.
IC IVAU{, <Xt>}

```
if PSTATE.EL == EL0 then
    if SCTLR_EL1.UCI == '0' then
```

```
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL2 then
        AArch64.IC(X[t, 64], CacheOpScope_PoU);
```

A.2.4.59 DC CVAC, Data or unified Cache line Clean by VA to PoC

Clean data cache by address to Point of Coherency.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-254: AARCH64_DC_CVAC bit assignments

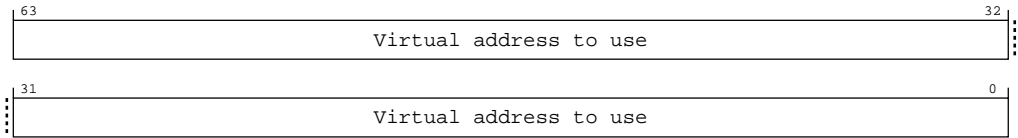


Table A-654: DC CVAC bit descriptions

Bits	Name	Description	Reset
[63:0]	VA	Virtual address to use. No alignment restrictions apply to this VA.	64 { x }

Access

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in *MMU faults generated by cache maintenance operations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see *The data cache maintenance instruction (DC)* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

DC CVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1010	0b001

Accessibility

If ELO access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.
DC CVAC, <Xt>

```

if PSTATE.EL == EL0 then
    if SCTL_EL1.UCI == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoC);

```

A.2.4.60 DC CVAU, Data or unified Cache line Clean by VA to PoU

Clean data cache by address to Point of Unification.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-255: AARCH64_DC_CVAU bit assignments

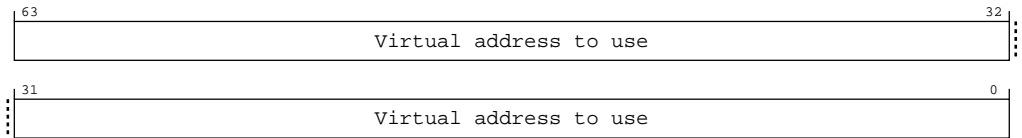


Table A-656: DC CVAU bit descriptions

Bits	Name	Description	Reset
[63:0]	VA	Virtual address to use. No alignment restrictions apply to this VA.	64 {x}

Access

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in *MMU faults generated by cache maintenance operations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see *The data cache maintenance instruction (DC)* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

DC CVAU, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1011	0b001

Accessibility

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.
DC CVAU, <Xt>

```
if PSTATE.EL == EL0 then
    if SCTLR_EL1.UCI == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoU);
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoU);
    elsif PSTATE.EL == EL2 then
```

```
AArch64.DC (X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoU);
```

A.2.4.61 DC CVAP, Data or unified Cache line Clean by VA to PoP

Clean data cache by address to Point of Persistence.

If the memory system does not identify a Point of Persistence, then this instruction behaves as a DC CVAC.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-256: AARCH64_DC_CVAP bit assignments

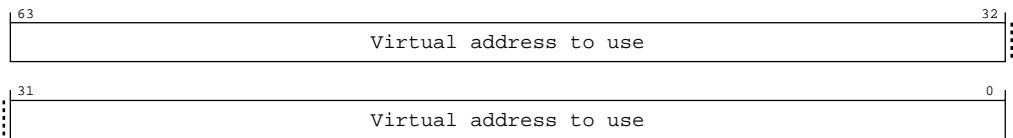


Table A-658: DC CVAP bit descriptions

Bits	Name	Description	Reset
[63:0]	VA	Virtual address to use. No alignment restrictions apply to this VA.	64 {x}

Access

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in *MMU faults generated by cache maintenance operations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see *The data cache maintenance instruction (DC)* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

DC CVAP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1100	0b001

Accessibility

If ELO access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.
DC CVAP, <Xt>

```

if PSTATE.EL == EL0 then
    if SCTL_EL1.UCI == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoP);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoP);

```

A.2.4.62 DC CVADP, Data or unified Cache line Clean by VA to PoDP

Clean data cache by address to Point of Deep Persistence.

If the memory system does not identify a Point of Deep Persistence, then this instruction behaves as a DC CVAP.

Configurations

This register is present only when IsFeatureImplemented(FEAT_DPB2). Otherwise, direct accesses to DC CVADP are UNDEFINED.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-257: AARCH64_DC_CVADP bit assignments

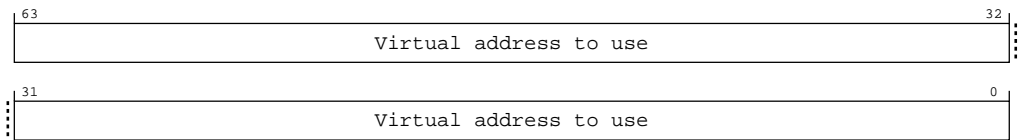


Table A-660: DC CVADP bit descriptions

Bits	Name	Description	Reset
[63:0]	VA	Virtual address to use. No alignment restrictions apply to this VA.	64 {x}

Access

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in *MMU faults generated by cache maintenance operations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see *The data cache maintenance instruction (DC)* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

DC CVADP, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1101	0b001

Accessibility

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.
DC CVADP, <Xt>

```
if PSTATE.EL == EL0 then
    if SCTLR_EL1.UCI == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoDP);
    elsif PSTATE.EL == EL2 then
```

```
AArch64.DC (X[t, 64], CacheType_Data, CacheOp_Clean, CacheOpScope_PoDP);
```

A.2.4.63 DC CIVAC, Data or unified Cache line Clean and Invalidate by VA to PoC

Clean and Invalidate data cache by address to Point of Coherency.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-258: AARCH64_DC_CIVAC bit assignments

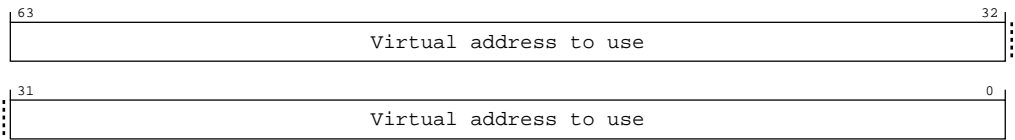


Table A-662: DC CIVAC bit descriptions

Bits	Name	Description	Reset
[63:0]	VA	Virtual address to use. No alignment restrictions apply to this VA.	64 { x }

Access

If ELO access is enabled, when executed at ELO, the instruction may generate a Permission fault, subject to the constraints described in *MMU faults generated by cache maintenance operations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see *The data cache maintenance instruction (DC)* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

DC CIVAC, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b1110	0b001

Accessibility

If ELO access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.
DC CIVAC, <Xt>

```
if PSTATE.EL == EL0 then
    if SCTL_EL1.UCI == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate,
                CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate,
                CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t, 64], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);
```

A.2.4.64 AT S1E2R, Address Translate Stage 1 EL2 Read

Performs stage 1 address translation as defined for EL2, with permissions as if reading from the given virtual address.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-259: AARCH64_AT_S1E2R bit assignments

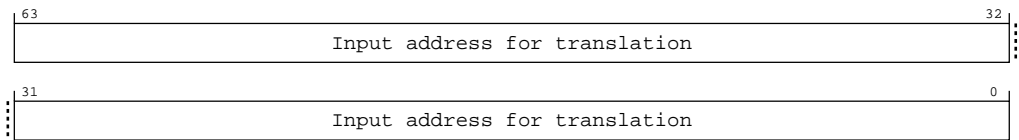


Table A-664: AT S1E2R bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access
AT S1E2R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b000

Accessibility
AT S1E2R, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL2, ATAccess_Read);
```

A.2.4.65 AT S1E2W, Address Translate Stage 1 EL2 Write

Performs stage 1 address translation as defined for EL2, with permissions as if writing to the given virtual address.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-260: AARCH64_AT_S1E2W bit assignments

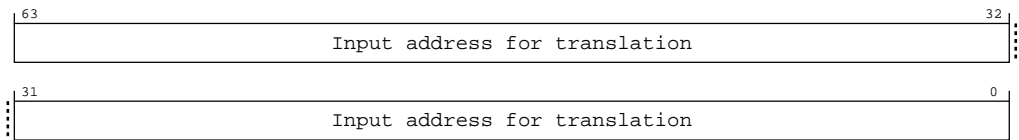


Table A-666: AT S1E2W bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access
AT S1E2W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b001

Accessibility
AT S1E2W, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.AT(X[t, 64], TranslationStage_1, EL2, ATAccess_Write);
```

A.2.4.66 AT S12E1R, Address Translate Stages 1 and 2 EL1 Read

Performs stage 1 and 2 address translation, with permissions as if reading from the given virtual address from EL1, using the EL1&O translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-261: AARCH64_AT_S12E1R bit assignments

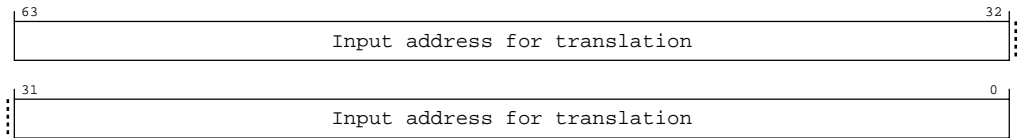


Table A-668: AT S12E1R bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access
AT S12E1R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b100

Accessibility
AT S12E1R, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.<DC,VM> == '00' then
        AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Read);
    else
        AArch64.AT(X[t, 64], TranslationStage_12, EL1, ATAccess_Read);
```

A.2.4.67 AT S12E1W, Address Translate Stages 1 and 2 EL1 Write

Performs stage 1 and 2 address translation, with permissions as if writing to the given virtual address from EL1, using the EL1&O translation regime.

Configurations
This register is available in all configurations.

Attributes
Width
64
Functional group
System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-262: AARCH64_AT_S12E1W bit assignments

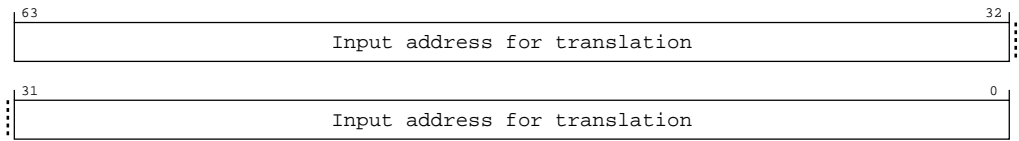


Table A-670: AT S12E1W bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S12E1W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b101

Accessibility

AT S12E1W, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.<DC,VM> == '00' then
        AArch64.AT(X[t, 64], TranslationStage_1, EL1, ATAccess_Write);
    else
        AArch64.AT(X[t, 64], TranslationStage_12, EL1, ATAccess_Write);
```

A.2.4.68 AT S12E0R, Address Translate Stages 1 and 2 ELO Read

Performs stage 1 and 2 address translations from ELO, with permissions as if reading from the given virtual address from ELO, using the EL1&0 translation regime.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group
System instructions

Access type
See bit descriptions

Bit descriptions

Figure A-263: AARCH64_AT_S12EOR bit assignments

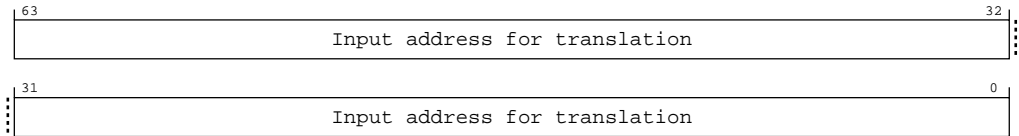


Table A-672: AT S12EOR bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access
AT S12EOR, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b110

Accessibility
AT S12EOR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.<DC,VM> == '00' then
        AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Read);
    else
        AArch64.AT(X[t, 64], TranslationStage_12, EL0, ATAccess_Read);
```

A.2.4.69 AT S12EOW, Address Translate Stages 1 and 2 ELO Write

Performs stage 1 and 2 address translations from ELO, with permissions as if writing to the given virtual address from ELO, using the EL1&0 translation regime.

Configurations
This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-264: AARCH64_AT_S12E0W bit assignments

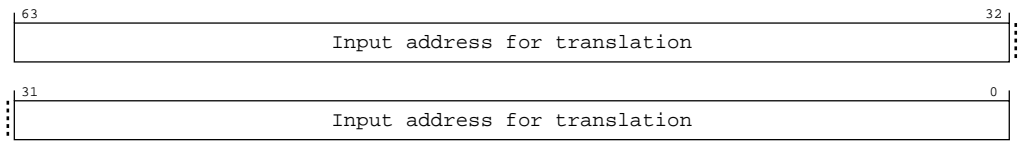


Table A-674: AT S12E0W bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Input address for translation. The resulting address can be read from the AArch64-PAR_EL1.	64 {x}

Access

AT S12E0W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b111

Accessibility

AT S12E0W, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.<DC,VM> == '00' then
        AArch64.AT(X[t, 64], TranslationStage_1, EL0, ATAccess_Write);
    else
        AArch64.AT(X[t, 64], TranslationStage_12, EL0, ATAccess_Write);
```

A.2.4.70 TLBI IPAS2E1IS, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk.
- The entry would be required to translate the specified IPA using the EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-265: AARCH64_TLBI_IPAS2E1IS bit assignments

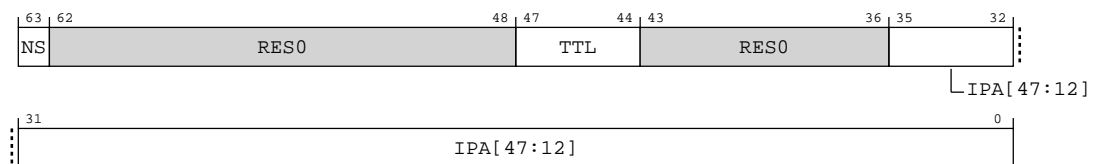


Table A-676: TLBI IPAS2E1IS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
[47:44]	TTL	Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated. 00xx No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0 . 01xx The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3. 10xx The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Reserved. Treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3. 11xx The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3. If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.	xxxx
[43:36]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[35:0]	IPA[47:12]	Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0 .	36 {x}

Access

TLBI IPAS2E1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b001

Accessibility

TLBI IPAS2E1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.71 TLBI RIPAS2E1IS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be required to translate any IPA in the specified address range using the EL1&O translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.

- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-266: AARCH64_TLBI_RIPAS2E1IS bit assignments

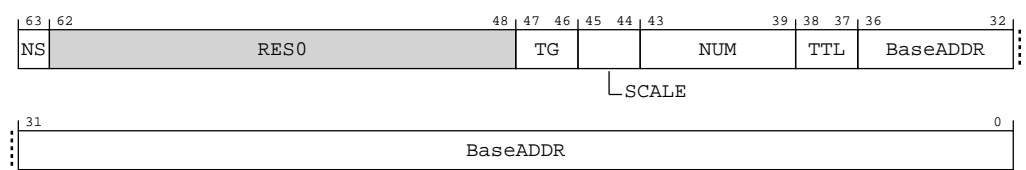


Table A-678: TLBI RIPAS2E1IS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RIPAS2E1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b010

Accessibility

TLBI RIPAS2E1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.72 TLBI IPAS2LE1IS, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the final level of the translation table walk.
- The entry would be required to translate the specified IPA using the EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-267: AARCH64_TLBI_IPAS2LE1IS bit assignments

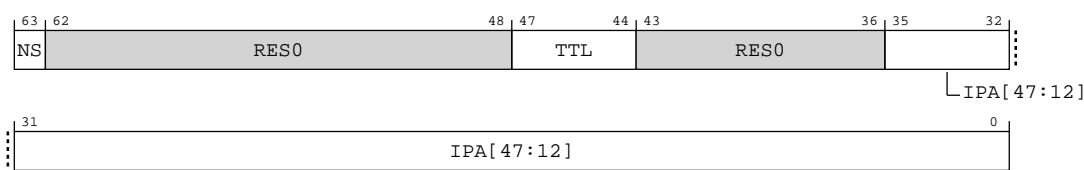


Table A-680: TLBI IPAS2LE1IS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:36]	RES0	Reserved	RES0
[35:0]	IPA[47:12]	Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0 .	36 {x}

Access

TLBI IPAS2LE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b101

Accessibility

TLBI IPAS2LE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.73 TLBI IPAS2LE1IS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- The entry would be required to translate any IPA in the specified address range using the EL1&O translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 00000000000000.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-268: AARCH64_TLBI_RIPAS2LE1IS bit assignments

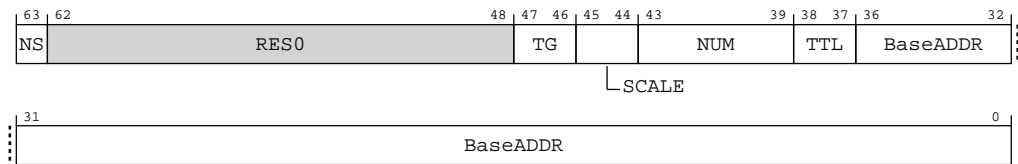


Table A-682: TLBI RIPAS2LE1IS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RIPAS2LE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0000	0b110

Accessibility

TLBI RIPAS2LE1IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.74 TLBI ALLE2OS, TLB Invalidate All, EL2, Outer Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

- The entry would be required to translate any address using the EL2&O or EL2 translation regime.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE2OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_ALL(SecurityStateAtEL(EL2), Regime_EL2, Shareability_OSH,
        TLBI_AllAttr);
```

A.2.4.75 TLBI VAE2OS, TLB Invalidate by VA, EL2, Outer Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.

- The entry would be required to translate the specified VA using the EL2 translation regime for the Security state.
- The entry is from any level of the translation table walk.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-269: AARCH64_TLBI_VAE2OS bit assignments

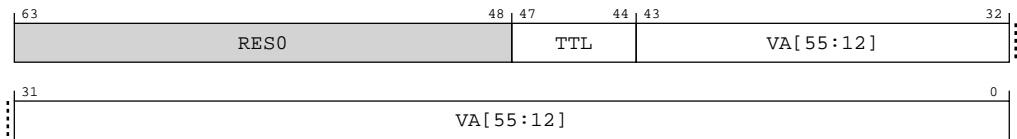


Table A-685: TLBI VAE2OS bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b001

Accessibility

TLBI VAE2OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
```


A.2.4.76 TLBI ALLE1OS, TLB Invalidate All, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL1&O translation regime.

The invalidation applies to entries with any VMID.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.



Note

For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b100

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_ALL(SecurityStateAtEL(EL1), Regime_EL10, Shareability_OSH,
        TLBI_AllAttr);

```

A.2.4.77 TLBI VALE2OS, TLB Invalidate by VA, Last level, EL2, Outer Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA

using the EL2 translation regime for the Security state.

- The entry is from the final level of the translation table walk.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group
System instructions

Access type
See bit descriptions

Bit descriptions

Figure A-270: AARCH64_TLBI_VALE2OS bit assignments

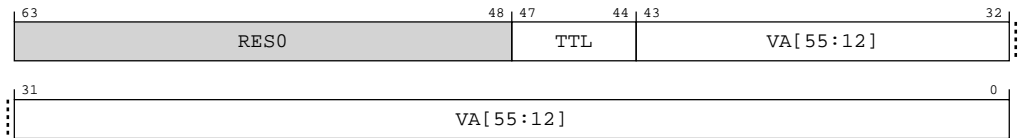


Table A-688: TLBI VALE2OS bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VALE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b101

Accessibility

TLBI VALE2OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
```


A.2.4.78 TLBI VMALLS12E1OS, TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Outer Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL1&O translation regime.
- The entry would be used with the current VMID.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.


When



Note

a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.



Note

For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLS12E1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0001	0b110

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLS12E1OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI VMALLS12(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBI_AllAttr);
```

A.2.4.79 TLBI RVAE2IS, TLB Range Invalidate by VA, EL2, Inner Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any VA

in the range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$

using the EL2 translation regime for the Security state.

- The entry is from any level of the translation table walk.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-271: AARCH64_TLBI_RVAE2IS bit assignments

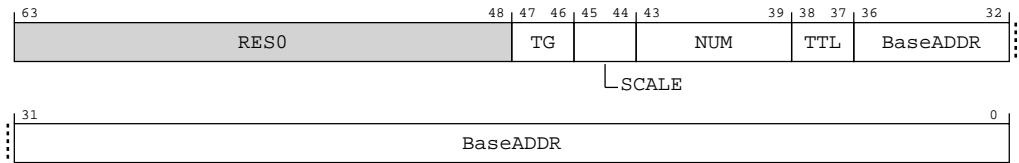


Table A-691: TLBI RVAE2IS bit descriptions

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

Bits	Name	Description	Reset
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVAE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0010	0b001

Accessibility

TLBI RVAE2IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.80 TLBI RVAE2IS, TLB Range Invalidate by VA, Last level, EL2, Inner Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any VA

in the range determined by the formula $[\text{BaseADDR} (5 * \text{SCALE} + 1) * \text{Translation_Granule_Size}]$

using the EL2 translation regime for the Security state.

- The entry is from the final level of the translation table walk.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-272: AARCH64_TLBI_RVALE2IS bit assignments

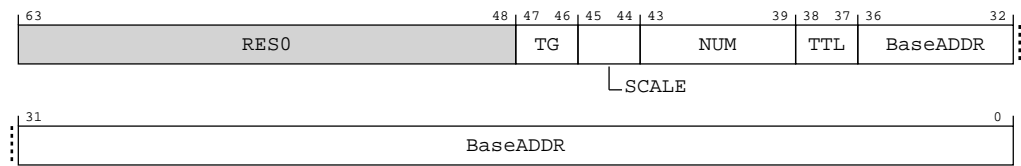


Table A-693: TLBI RVALE2IS bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVALE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0010	0b101

Accessibility

TLBI RVALE2IS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_ISH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.81 TLBI ALLE2IS, TLB Invalidate All, EL2, Inner Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.

- The entry would be required to translate any address using the EL2&0 or EL2 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE2IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_ALL(SecurityStateAtEL(EL2), Regime_EL2, Shareability_ISH,
        TLBI_AllAttr);
```

A.2.4.82 TLBI VAE2IS, TLB Invalidate by VA, EL2, Inner Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.

- The entry would be required to translate the specified VA using the EL2 translation regime for the Security state.
- The entry is from any level of the translation table walk.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-273: AARCH64_TLBI_VAE2IS bit assignments

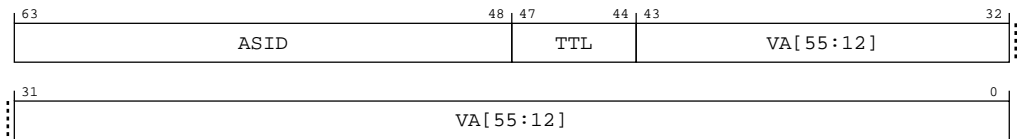


Table A-696: TLBI VAE2IS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16{x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b001

Accessibility

TLBI VAE2IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
```


A.2.4.83 TLBI ALLE1IS, TLB Invalidate All, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL1&O translation regime.

The invalidation applies to entries with any VMID.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.



Note

For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b100

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE1IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_ALL(SecurityStateAtEL(EL1), Regime_EL10, Shareability_ISH,
        TLBI_AllAttr);
```

A.2.4.84 TLBI VALE2IS, TLB Invalidate by VA, Last level, EL2, Inner Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA

using the EL2 translation regime for the Security state.

- The entry is from the final level of the translation table walk.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group
System instructions

Access type
See bit descriptions

Bit descriptions

Figure A-274: AARCH64_TLBI_VALE2IS bit assignments

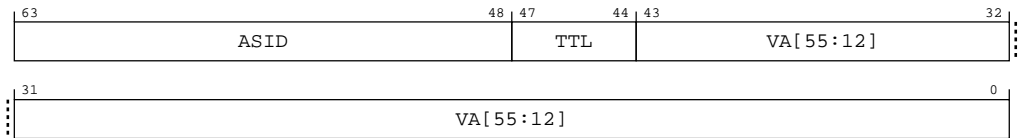


Table A-699: TLBI VALE2IS bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VALE2IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b101

Accessibility

TLBI VALE2IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_ISH,
    TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);
```


A.2.4.85 TLBI VMALLS12E1IS, TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.


From Armv8.4, when



Note

a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.



Note

For the EL1&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLS12E1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b110

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLS12E1IS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI VMALLS12(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_ISH, TLBI_AllAttr);
```

A.2.4.86 TLBI IPAS2E1OS, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk.
- The entry would be required to translate the specified IPA using the EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-275: AARCH64_TLBI_IPAS2E1OS bit assignments

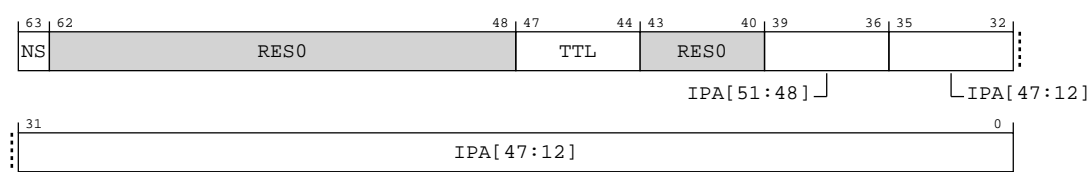


Table A-702: TLBI IPAS2E1OS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:40]	RES0	Reserved	RES0
[39:36]	IPA[51:48]	Extension to IPA[47:12]. For more information, see IPA[47:12].	xxxx
[35:0]	IPA[47:12]	Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0 .	36{x}

Access

TLBI IPAS2E1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b000

Accessibility

TLBI IPAS2E1OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
```

A.2.4.87 TLBI IPAS2E1, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk.
- The entry would be required to translate the specified IPA using the EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-276: AARCH64_TLBI_IPAS2E1 bit assignments

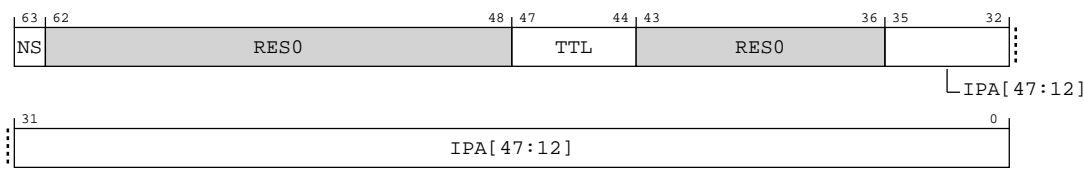


Table A-704: TLBI IPAS2E1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:36]	RES0	Reserved	RES0
[35:0]	IPA[47:12]	Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0 .	36 {x}

Access

TLBI IPAS2E1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b001

Accessibility

TLBI IPAS2E1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.88 TLBI RIPAS2E1, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be required to translate any IPA in the specified address range using the EL1&O translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-277: AARCH64_TLBI_RIPAS2E1 bit assignments

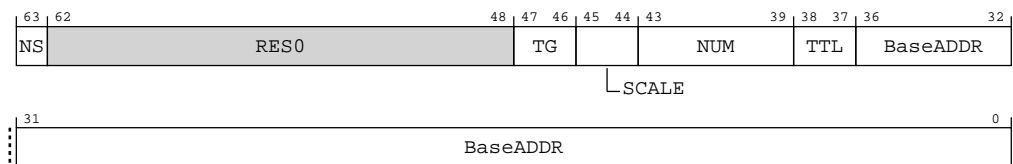


Table A-706: TLBI RIPAS2E1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RIPAS2E1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b010

Accessibility

TLBI RIPAS2E1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBI_Level_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.89 TLBI RIPAS2E1OS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be required to translate any IPA in the specified address range using the EL1&O translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $TTL == 01$ and $BaseADDR[29:12]$ is not equal to 000000000000000000.
 - If $TTL == 10$ and $BaseADDR[20:12]$ is not equal to 0000000000.
- For the 16K translation granule:
 - If $TTL == 10$ and $BaseADDR[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $TTL == 01$ and $BaseADDR[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $TTL == 10$ and $BaseADDR[28:16]$ is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-278: AARCH64_TLBI_RIPAS2E1OS bit assignments

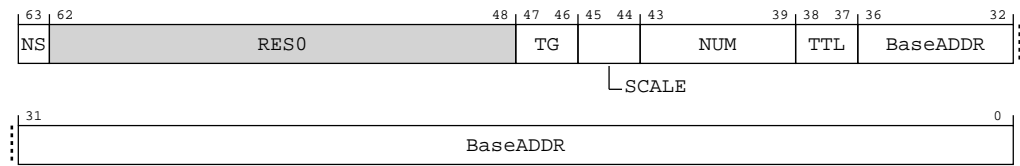


Table A-708: TLBI RIPAS2E1OS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}

Bits	Name	Description	Reset
[38:37]	TTL	<div>TTL Level hint. The TTL hint is only guaranteed to invalidate:<ul style="list-style-type: none">Non-leaf-level entries in the range up to but not including the level described by the TTL hint.Leaf-level entries in the range that match the level described by the TTL hint.</div> <div>0b00 The entries in the range can be using any level for the translation table entries.</div> <div>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries. When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</div> <div>0b10 The TTL hint indicates level 2.</div> <div>0b11 The TTL hint indicates level 3.</div>	xx
[36:0]	BaseADDR	<div>The starting address for the range of the maintenance instruction.</div> <div>When using a 4KB translation granule, this field is BaseADDR[48:12].</div> <div>When using a 16KB translation granule, this field is BaseADDR[50:14].</div> <div>When using a 64KB translation granule, this field is BaseADDR[52:16].</div>	37{x}

Access

TLBI RIPAS2E1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b011

Accessibility

TLBI RIPAS2E1OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
```

A.2.4.90 TLBI IPAS2LE1OS, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the final level of the translation table walk.
- The entry would be required to translate the specified IPA using the EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-279: AARCH64_TLBI_IPAS2LE1OS bit assignments

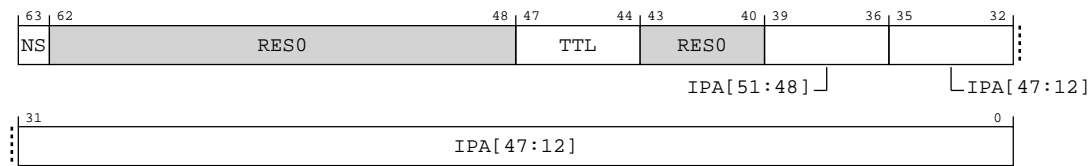


Table A-710: TLBI IPAS2LE1OS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
[47:44]	TTL	Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated. 00xx No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0 . 01xx The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3. 10xx The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Reserved. Treat as if TTL<3:2> is 0b00. 0b10 : Level 2. 0b11 : Level 3. 11xx The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as: 0b00 : Reserved. Treat as if TTL<3:2> is 0b00. 0b01 : Level 1. 0b10 : Level 2. 0b11 : Level 3. If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.	xxxx
[43:40]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[39:36]	IPA[51:48]	Extension to IPA[47:12]. For more information, see IPA[47:12].	xxxx
[35:0]	IPA[47:12]	Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0 .	36{x}

Access

TLBI IPAS2LE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b100

Accessibility

TLBI IPAS2LE1OS{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
```

A.2.4.91 TLBI IPAS2LE1, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the final level of the translation table walk.
- The entry would be required to translate the specified IPA using the EL1&0 translation regime.
- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to the PE that executes this System instruction.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-280: AARCH64_TLBI_IPAS2LE1 bit assignments

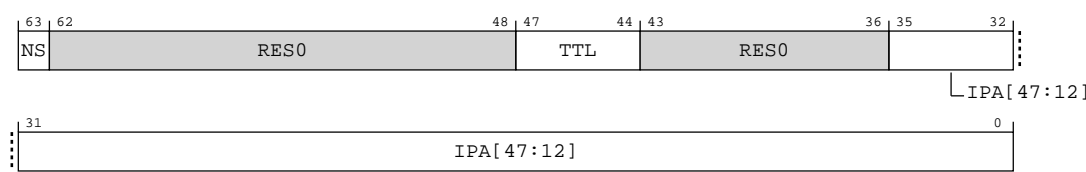


Table A-712: TLBI IPAS2LE1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:36]	RES0	Reserved	RES0
[35:0]	IPA[47:12]	Bits[47:12] of the intermediate physical address to match. For implementations with fewer than 48 bits, the upper bits of this field are RES0 .	36 {x}

Access

TLBI IPAS2LE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b101

Accessibility

TLBI IPAS2LE1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_IPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.92 TLBI RIPAS2LE1, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- The entry would be required to translate any IPA in the specified address range using the EL1&O translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation only applies to the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-281: AARCH64_TLBI_RIPAS2LE1 bit assignments

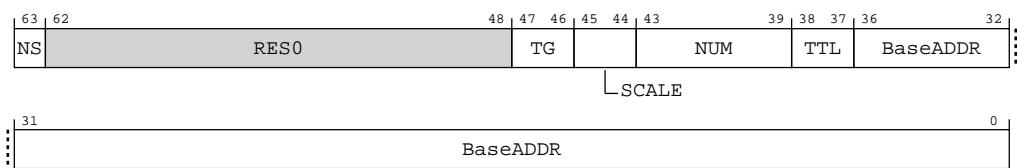


Table A-714: TLBI RIPAS2LE1 bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 { x }

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RIPAS2LE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b110

Accessibility

TLBI RIPAS2LE1{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBILevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.93 TLBI RIPAS2LE1OS, TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the leaf level of the translation table walk, indicated by the TTL hint.
- The entry would be required to translate any IPA in the specified address range using the EL1&O translation regime.
- The entry would be used with the current VMID.
- The entry is within the address range determined by the formula $[\text{BaseADDR} (5 * \text{SCALE} + 1) * \text{Translation_Granule_Size}]$.

When

a TLB maintenance instruction is generated to the Secure EL1&O translation regime and is defined to pass a VMID argument, then:

- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&O translation of a System MMU in the same required shareability domain with a VMID of 0.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $\text{TTL} = 01$ and $\text{BaseADDR}[29:12]$ is not equal to 000000000000000000.
 - If $\text{TTL} = 10$ and $\text{BaseADDR}[20:12]$ is not equal to 0000000000.
- For the 16K translation granule:
 - If $\text{TTL} = 10$ and $\text{BaseADDR}[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $\text{TTL} = 01$ and $\text{BaseADDR}[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $\text{TTL} = 10$ and $\text{BaseADDR}[28:16]$ is not equal to 0000000000000000.

For more information about the architectural requirements for this System instruction, see *Invalidation of TLB entries from stage 2 translations* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-282: AARCH64_TLBI_RIPAS2LE1OS bit assignments

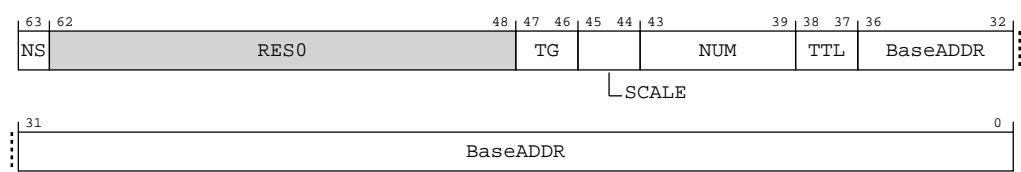


Table A-716: TLBI RIPAS2LE1OS bit descriptions

Bits	Name	Description	Reset
[63]	NS	Not Secure. Specifies the IPA space. 0b0 IPA is in the Secure IPA space. 0b1 IPA is in the Non-secure IPA space.	x
[62:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 { x }

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RIPAS2LE1OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b111

Accessibility

TLBI RIPAS2LE1OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RIPAS2(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_OSH, TLBI_Level_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.94 TLBI RVAE2OS, TLB Range Invalidate by VA, EL2, Outer Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any VA

in the range determined by the formula $[\text{BaseADDR} (5 * \text{SCALE} + 1) * \text{Translation_Granule_Size}]$

using the EL2 translation regime for the Security state.

- The entry is from any level of the translation table walk.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $\text{TTL} == 01$ and $\text{BaseADDR}[29:12]$ is not equal to 000000000000000000.
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[20:12]$ is not equal to 0000000000.
- For the 16K translation granule:
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $\text{TTL} == 01$ and $\text{BaseADDR}[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[28:16]$ is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-283: AARCH64_TLBI_RVAE2OS bit assignments

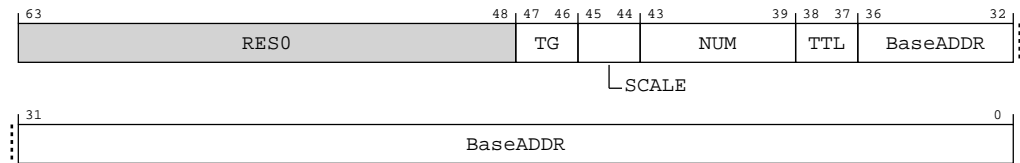


Table A-718: TLBI RVAE2OS bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx

Bits	Name	Description	Reset
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVAE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0101	0b001

Accessibility

TLBI RVAE2OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.95 TLBI RVAE2OS, TLB Range Invalidate by VA, Last level, EL2, Outer Shareable

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any VA

in the range determined by the formula $[\text{BaseADDR} (5 \times \text{SCALE} + 1) \times \text{Translation_Granule_Size}]$

using the EL2 translation regime for the Security state.

- The entry is from the final level of the translation table walk.

The invalidation applies to all PEs in the same Outer Shareable shareability domain as the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $\text{TTL} = 01$ and $\text{BaseADDR}[29:12]$ is not equal to 0000000000000000.

- If TTL==10 and BaseADDR[20:12] is not equal to 000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 00000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-284: AARCH64_TLBI_RVALE2OS bit assignments

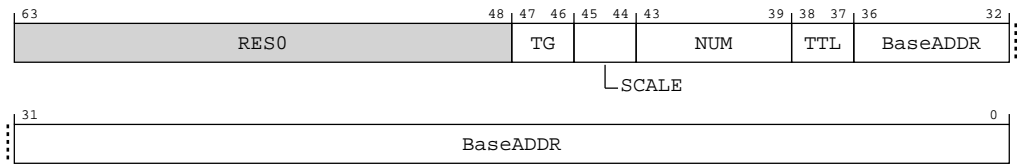


Table A-720: TLBI RVALE2OS bit descriptions

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

Bits	Name	Description	Reset
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37 {x}

Access

TLBI RVALE2OS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0101	0b101

Accessibility

TLBI RVALE2OS{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_OSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.96 TLBI RVAE2, TLB Range Invalidate by VA, EL2

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any VA

in the range determined by the formula $[BaseADDR (5 * SCALE + 1) * Translation_Granule_Size]$

using the EL2 translation regime for the Security state.

- The entry is from any level of the translation table walk.

The invalidation applies to the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If TTL==01 and BaseADDR[29:12] is not equal to 00000000000000000000.
 - If TTL==10 and BaseADDR[20:12] is not equal to 0000000000.
- For the 16K translation granule:
 - If TTL==10 and BaseADDR[24:14] is not equal to 000000000000.
- For the 64K translation granule:
 - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group
System instructions

Access type
See bit descriptions

Bit descriptions

Figure A-285: AARCH64_TLBI_RVAE2 bit assignments

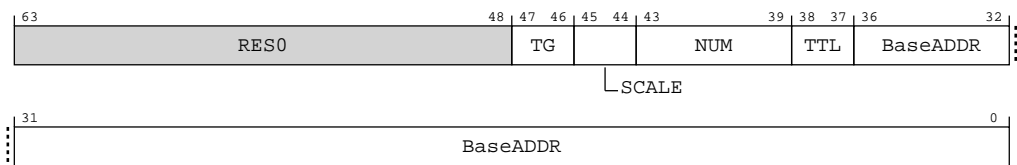


Table A-722: TLBI RVAE2 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	Translation granule size. 0b00 Reserved. 0b01 4K translation granule. 0b10 16K translation granule. 0b11 64K translation granule. The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 { x }

Bits	Name	Description	Reset
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVAE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0110	0b001

Accessibility

TLBI RVAE2{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);

```

A.2.4.97 TLBI RVALE2, TLB Range Invalidate by VA, Last level, EL2

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk up to the level indicated in the TTL hint.
- The entry would be used to translate any VA

in the range determined by the formula $[\text{BaseADDR} (5 * \text{SCALE} + 1) * \text{Translation_Granule_Size}]$

using the EL2 translation regime for the Security state.

- The entry is from the final level of the translation table walk.

The invalidation applies to the PE that executes this System instruction.

For translation table entry, the range of addresses invalidated is **UNPREDICTABLE** when:

- For the 4K translation granule:
 - If $\text{TTL} == 01$ and $\text{BaseADDR}[29:12]$ is not equal to 000000000000000000.
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[20:12]$ is not equal to 0000000000.
- For the 16K translation granule:
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[24:14]$ is not equal to 000000000000.
- For the 64K translation granule:
 - If $\text{TTL} == 01$ and $\text{BaseADDR}[41:16]$ is not equal to 00000000000000000000000000000000.
 - If $\text{TTL} == 10$ and $\text{BaseADDR}[28:16]$ is not equal to 0000000000000000.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-286: AARCH64_TLBI_RVALE2 bit assignments

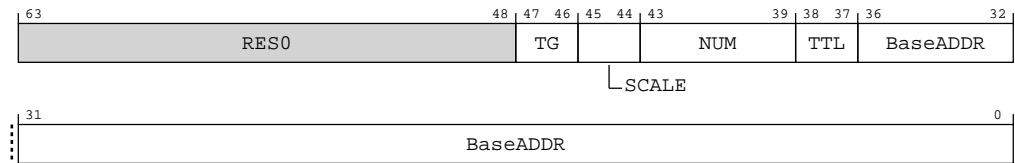


Table A-724: TLBI RVALE2 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	RES0
[47:46]	TG	<p>Translation granule size.</p> <p>0b00 Reserved.</p> <p>0b01 4K translation granule.</p> <p>0b10 16K translation granule.</p> <p>0b11 64K translation granule.</p> <p>The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.</p>	xx
[45:44]	SCALE	The exponent element of the calculation that is used to produce the upper range.	xx
[43:39]	NUM	The base element of the calculation that is used to produce the upper range.	5 {x}
[38:37]	TTL	<p>TTL Level hint. The TTL hint is only guaranteed to invalidate:</p> <ul style="list-style-type: none"> Non-leaf-level entries in the range up to but not including the level described by the TTL hint. Leaf-level entries in the range that match the level described by the TTL hint. <p>0b00 The entries in the range can be using any level for the translation table entries.</p> <p>0b01 When using a 4KB or 64KB translation granule, all entries to invalidate are Level 1 translation table entries.</p> <p>When using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00.</p> <p>0b10 The TTL hint indicates level 2.</p> <p>0b11 The TTL hint indicates level 3.</p>	xx

Bits	Name	Description	Reset
[36:0]	BaseADDR	<p>The starting address for the range of the maintenance instruction.</p> <p>When using a 4KB translation granule, this field is BaseADDR[48:12].</p> <p>When using a 16KB translation granule, this field is BaseADDR[50:14].</p> <p>When using a 64KB translation granule, this field is BaseADDR[52:16].</p>	37{x}

Access

TLBI RVALE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0110	0b101

Accessibility

TLBI RVALE2{, <Xt>}

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_RVA(SecurityStateAtEL(EL2), Regime_EL2, VMID[], Shareability_NSH,
        TLBIlevel_Last, TLBI_AllAttr, X[t, 64]);

```

A.2.4.98 TLBI ALLE2, TLB Invalidate All, EL2

If EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be required to translate any address using the EL2&O or EL2 translation regime.

The invalidation only applies to the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b000

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE2{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_ALL(SecurityStateAtEL(EL2), Regime_EL2, Shareability_NSH,
        TLBI_AllAttr);
```

A.2.4.99 TLBI VAE2, TLB Invalidate by VA, EL2

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be required to translate the specified VA

using the EL2 translation regime for the Security state.

- The entry is from any level of the translation table walk.

The invalidation applies to the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type

See bit descriptions

Bit descriptions

Figure A-287: AARCH64_TLBI_VAE2 bit assignments

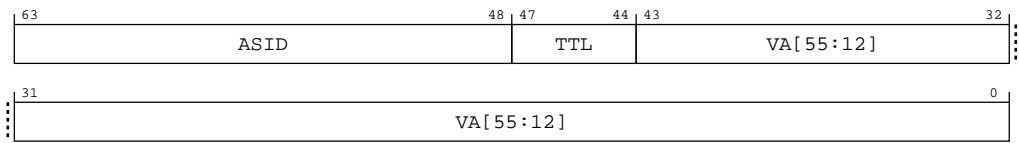


Table A-727: TLBI VAE2 bit descriptions

Bits	Name	Description	Reset
[63:48]	ASID	ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction. Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field. If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.	16 {x}

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VAE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b001

Accessibility

TLBI VAE2{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBIlevel_Any, TLBI_AllAttr, X[t, 64]);
```


A.2.4.100 TLBI ALLE1, TLB Invalidate All, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL1&O translation regime.

The invalidation applies to entries with any VMID.

The invalidation only applies to the PE that executes this System instruction.



Note

For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b100

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI ALLE1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_ALL(SecurityStateAtEL(EL1), Regime_EL10, Shareability_NSH,
        TLBI_AllAttr);
```

A.2.4.101 TLBI VALE2, TLB Invalidate by VA, Last level, EL2

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified VA

using the EL2 translation regime for the Security state.

- The entry is from the final level of the translation table walk.

The invalidation applies to the PE that executes this System instruction.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access type
See bit descriptions

Bit descriptions

Figure A-288: AARCH64_TLBI_VALE2 bit assignments

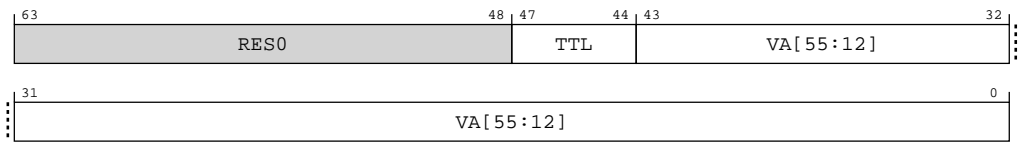


Table A-730: TLBI VALE2 bit descriptions

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

Bits	Name	Description	Reset
[47:44]	TTL	<p>Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.</p> <p>00xx</p> <p>No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is RES0.</p> <p>01xx</p> <p>The entry comes from a 4KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>10xx</p> <p>The entry comes from a 16KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>11xx</p> <p>The entry comes from a 64KB translation granule. The level of walk for the leaf level 0bxx is encoded as:</p> <p>0b00 : Reserved. Treat as if TTL<3:2> is 0b00.</p> <p>0b01 : Level 1.</p> <p>0b10 : Level 2.</p> <p>0b11 : Level 3.</p> <p>If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.</p>	xxxx
[43:0]	VA[55:12]	<p>Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.</p> <p>The treatment of the low-order bits of this field depends on the translation granule size, as follows:</p> <ul style="list-style-type: none"> Where a 4KB translation granule is being used, all bits are valid and used for the invalidation. Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction. Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction. 	44 {x}

Access

TLBI VALE2{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b101

Accessibility

TLBI VALE2{, <Xt>}


```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI_VA(SecurityStateAtEL(EL2), Regime_EL2, VMID_NONE, Shareability_NSH,
        TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
```

A.2.4.102 TLBI VMALLS12E1, TLB Invalidate by VMID, All at Stage 1 and 2, EL1

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL1&O translation regime.
- The entry would be used with the current VMID.

The invalidation applies to the PE that executes this System instruction.



Note

For the EL1&O translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

System instructions

Access

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLS12E1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0111	0b110

Accessibility

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is **CONSTRAINED UNPREDICTABLE**, and the implemented behavior is:

- The instruction behaves as if the Rt field is set to 0b11111.

TLBI VMALLS12E1{, <Xt>}

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    AArch64.TLBI VMALLS12(SecurityStateAtEL(EL1), Regime_EL10, VMID[],
        Shareability_NSH, TLBI_AllAttr);
```

A.2.5 AArch64 Debug register description

This section includes the register descriptions for all Debug registers in the Cortex®-R82 processor.

A.2.5.1 OSDTRRX_EL1, OS Lock Data Transfer Register, Receive

Used for save and restore of AArch64-DBGDTRRX_EL0. It is a component of the Debug Communications Channel.

Configurations

This register is available in all configurations.

Attributes

Width

64

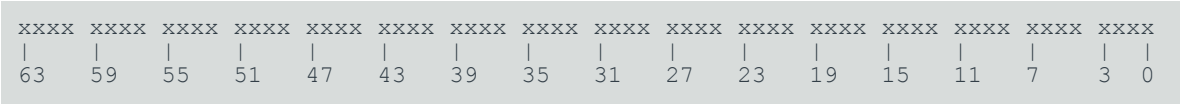
Functional group

Debug registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-289: AARCH64_OSDTRRX_EL1 bit assignments

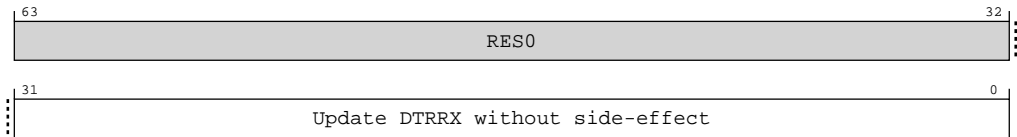


Table A-733: OSDTRRX_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	None	Update DTRRX without side-effect. Writes to this register update the value in DTRRX and do not change RXfull. Reads of this register return the last value written to DTRRX and do not change RXfull. For the full behavior of the Debug Communications Channel, see <i>The Debug Communication Channel and Instruction Transfer Register</i> in the Arm® Architecture Reference Manual for A-profile architecture .	32 {x}

Access

Arm deprecates reads and writes of OSDTRRX_EL1 when the OS Lock is unlocked.

MRS <Xt>, OSDTRRX_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b010

MSR OSDTRRX_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0000	0b010

Accessibility

Arm deprecates reads and writes of OSDTRRX_EL1 when the OS Lock is unlocked.

MRS <Xt>, OSDTRRX_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif Halted() then
    X[t, 64] = OSDTRRX_EL1;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = OSDTRRX_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = OSDTRRX_EL1;

```

MSR OSDTRRX_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif Halted() then
    OSDTRRX_EL1 = X[t, 64];
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        OSDTRRX_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    OSDTRRX_EL1 = X[t, 64];

```

A.2.5.2 MDCCINT_EL1, Monitor DCC Interrupt Enable Register

Enables interrupt requests to be signaled based on the DCC status flags.

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	x00x	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-290: AARCH64_MDCCINT_EL1 bit assignments

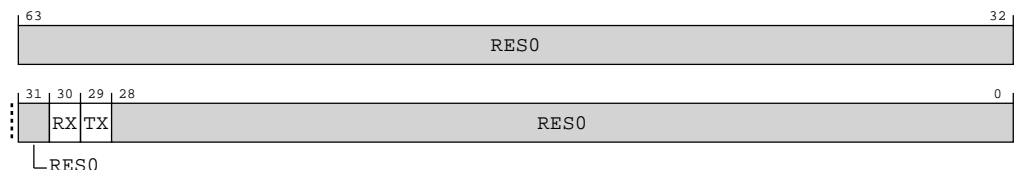


Table A-736: MDCCINT_EL1 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0
[30]	RX	DCC interrupt request enable control for DTRRX. Enables a common COMMIRQ interrupt request to be signaled based on the DCC status flags. 0b0 No interrupt request generated by DTRRX. 0b1 Interrupt request will be generated on RXfull == 1.	0b0
[29]	TX	DCC interrupt request enable control for DTRTX. Enables a common COMMIRQ interrupt request to be signaled based on the DCC status flags. 0b0 No interrupt request generated by DTRTX. 0b1 Interrupt request will be generated on TXfull == 0.	0b0
[28:0]	RES0	Reserved	RES0

Access

MRS <Xt>, MDCCINT_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b000

MSR MDCCINT_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b000

Accessibility

MRS <Xt>, MDCCINT_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif Halted() then
    X[t, 64] = MDCCINT_EL1;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MDCCINT_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MDCCINT_EL1;
```

MSR MDCCINT_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif Halted() then
    MDCCINT_EL1 = X[t, 64];
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        MDCCINT_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    MDCCINT_EL1 = X[t, 64];
```

A.2.5.3 MDSCR_EL1, Monitor Debug System Control Register

Main control register for the debug implementation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Debug registers

Access type

R

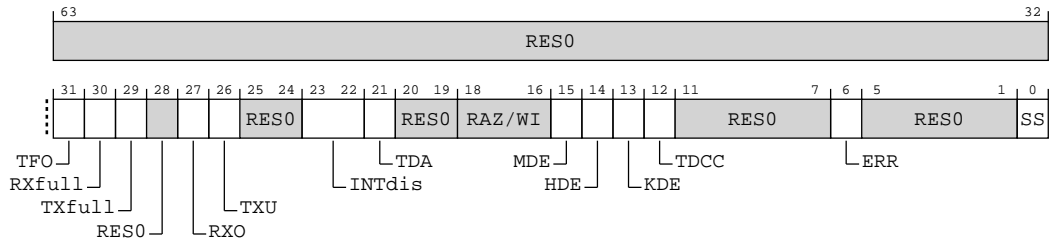
Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x000	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-291: AARCH64_MDSCR_EL1 bit assignments**Table A-739: MDSCR_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	TFO	Trace Filter override. Used for save/restore of ext-EDSCR.TFO. When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP. When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.TFO. Reads and writes of this bit are indirect accesses to ext-EDSCR.TFO. When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO	x
[30]	RXfull	Used for save/restore of ext-EDSCR.RXfull. When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP. When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.RXfull. Reads and writes of this bit are indirect accesses to ext-EDSCR.RXfull. When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO	x ²¹

²¹ The architected behavior of this field determines the value it returns after a reset.

Bits	Name	Description	Reset
[29]	TXfull	<p>Used for save/restore of ext-EDSCR.TXfull.</p> <p>When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP.</p> <p>When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.TXfull. Reads and writes of this bit are indirect accesses to ext-EDSCR.TXfull.</p> <p>When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW</p> <p>When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO</p>	x ²²
[28]	RES0	Reserved	RES0
[27]	R XO	<p>Used for save/restore of ext-EDSCR.R XO.</p> <p>When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP.</p> <p>When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.R XO. Reads and writes of this bit are indirect accesses to ext-EDSCR.R XO.</p> <p>When AArch64-OSLSR_EL1.OSLK == 1, if bits [27,6] of the value written to MDSCR_EL1 are {1,0}, that is, the R XO bit is 1 and the ERR bit is 0, the PE sets ext-EDSCR.{R XO,ERR} to UNKNOWN values.</p> <p>When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW</p> <p>When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO</p>	x ²³
[26]	TXU	<p>Used for save/restore of ext-EDSCR.TXU.</p> <p>When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP.</p> <p>When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.TXU. Reads and writes of this bit are indirect accesses to ext-EDSCR.TXU.</p> <p>When AArch64-OSLSR_EL1.OSLK == 1, if bits [26,6] of the value written to MDSCR_EL1 are {1,0}, that is, the TXU bit is 1 and the ERR bit is 0, the PE sets ext-EDSCR.{TXU,ERR} to UNKNOWN values.</p> <p>When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW</p> <p>When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO</p>	x ²⁴
[25:24]	RES0	Reserved	RES0

²² The architected behavior of this field determines the value it returns after a reset.

²³ The architected behavior of this field determines the value it returns after a reset.

²⁴ The architected behavior of this field determines the value it returns after a reset.

Bits	Name	Description	Reset
[23:22]	INTdis	Used for save/restore of ext-EDSCR.INTdis. When AArch64-OSLSR_EL1.OSLK == 0, and software must treat this bit as UNK/SBZP. When AArch64-OSLSR_EL1.OSLK == 1, this field holds the value of ext-EDSCR.INTdis. Reads and writes of this field are indirect accesses to ext-EDSCR.INTdis. When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO	xx ²⁵
[21]	TDA	Used for save/restore of ext-EDSCR.TDA. When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP. When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.TDA. Reads and writes of this bit are indirect accesses to ext-EDSCR.TDA. When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO	x ²⁶
[20:19]	RES0	Reserved	RES0
[18:16]	RAZ/WI	Reserved	RAZ/WI
[15]	MDE	Monitor debug events. Enable Breakpoint, Watchpoint, and Vector Catch exceptions. 0b0 Breakpoint, Watchpoint, and Vector Catch exceptions disabled. 0b1 Breakpoint, Watchpoint, and Vector Catch exceptions enabled.	x
[14]	HDE	Used for save/restore of ext-EDSCR.HDE. When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP. When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.HDE. Reads and writes of this bit are indirect accesses to ext-EDSCR.HDE. When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO	x ²⁷
[13]	KDE	Local (kernel) debug enable. If EL _D is using AArch64, enable debug exceptions within EL _D . Permitted values are: 0b0 Debug exceptions, other than Breakpoint Instruction exceptions, disabled within EL _D . 0b1 All debug exceptions enabled within EL _D .	x

²⁵ The architected behavior of this field determines the value it returns after a reset.²⁶ The architected behavior of this field determines the value it returns after a reset.²⁷ The architected behavior of this field determines the value it returns after a reset.

Bits	Name	Description	Reset
[12]	TDCC	<p>Traps EL0 accesses to the Debug Communication Channel (DCC) registers to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, from both Execution states, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, MRS or MSR accesses to the following DCC registers are trapped, reported using EC syndrome value 0x18: <ul style="list-style-type: none"> AArch64-MDCCSR_ELO. If not in Debug state, AArch64-DBGDTR_ELO, AArch64-DBGDTRTX_ELO, and AArch64-DBGDTRRX_ELO. <p>0b0 This control does not cause any instructions to be trapped.</p> <p>0b1 EL0 using AArch64: EL0 accesses to the AArch64 DCC registers are trapped.</p>	x
[11:7]	RES0	Reserved	RES0
[6]	ERR	<p>Used for save/restore of ext-EDSCR.ERR.</p> <p>When AArch64-OSLSR_EL1.OSLK == 0, software must treat this bit as UNK/SBZP.</p> <p>When AArch64-OSLSR_EL1.OSLK == 1, this bit holds the value of ext-EDSCR.ERR. Reads and writes of this bit are indirect accesses to ext-EDSCR.ERR.</p> <p>When AArch64-OSLSR_EL1.OSLK == '1' Access to this field is: RW</p> <p>When AArch64-OSLSR_EL1.OSLK == '0' Access to this field is: RO</p>	x ²⁸
[5:1]	RES0	Reserved	RES0
[0]	SS	<p>Software step control bit. If EL_D is using AArch64, enable Software step. Permitted values are:</p> <p>0b0 Software step disabled</p> <p>0b1 Software step enabled.</p>	x

Access

MRS <Xt>, MDSCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b010

MSR MDSCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0010	0b010

²⁸ The architected behavior of this field determines the value it returns after a reset.

Accessibility

MRS <Xt>, MDSCR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MDSCR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MDSCR_EL1;

```

MSR MDSCR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        MDSCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    MDSCR_EL1 = X[t, 64];

```

A.2.5.4 OSDTRTX_EL1, OS Lock Data Transfer Register, Transmit

Used for save/restore of AArch64-DBGDTRTX_EL0. It is a component of the Debug Communications Channel.

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-292: AARCH64_OSDTRTX_EL1 bit assignments

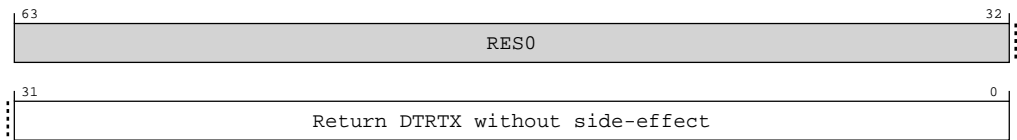


Table A-742: OSDTRTX_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	None	Return DTRTX without side-effect. Reads of this register return the value in DTRTX and do not change TXfull. Writes of this register update the value in DTRTX and do not change TXfull. For the full behavior of the Debug Communications Channel, see <i>The Debug Communication Channel and Instruction Transfer Register</i> in the <i>Arm® Architecture Reference Manual for A-profile architecture</i> .	32 {x}

Access

Arm deprecates reads and writes of OSDTRTX_EL1 when the OS Lock is unlocked.

MRS <Xt>, OSDTRTX_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b010

MSR OSDTRTX_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0011	0b010

Accessibility

Arm deprecates reads and writes of OSDTRTX_EL1 when the OS Lock is unlocked.

MRS <Xt>, OSDTRTX_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif Halted() then
    X[t, 64] = OSDTRTX_EL1;
```

```

elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = OSDTRTX_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = OSDTRTX_EL1;

```

MSR OSDTRTX_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif Halted() then
    OSDTRTX_EL1 = X[t, 64];
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        OSDTRTX_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    OSDTRTX_EL1 = X[t, 64];

```

A.2.5.5 OSECCR_EL1, OS Lock Exception Catch Control Register

Provides a mechanism for an operating system to access the contents of ext-EDECCR that are otherwise invisible to software, so it can save/restore the contents of ext-EDECCR over powerdown on behalf of the external debugger.

Configurations

If AArch64-OSLSR_EL1.OSLK == 0, then OSECCR_EL1 returns an UNKNOWN value on reads and ignores writes.

AArch64 register OSECCR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.1.9 EDECCR, External Debug Exception Catch Control Register](#) on page 1634 bits [31:0].

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	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

When AArch64-OSLSR_EL1.OSLK == '1'

Figure A-293: AARCH64_OSECCR_EL1 bit assignments

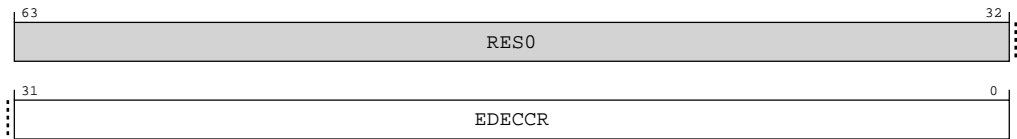


Table A-745: OSECCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	EDECCR	Used for save/restore to ext-EDECCR over powerdown. Reads or writes to this field are indirect accesses to ext-EDECCR.	32 {x}

Access

MRS <Xt>, OSECCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0110	0b010

MSR OSECCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0110	0b010

Accessibility

MRS <Xt>, OSECCR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif OSLSR_EL1.OSLK == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = OSECCR_EL1;
elsif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' then
        X[t, 64] = bits(64) UNKNOWN;
```

```
else
    X[t, 64] = OSECCR_EL1;
```

MSR OSECCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif OSLSR_EL1.OSLK == '0' then
        return;
    else
        OSECCR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' then
        return;
    else
        OSECCR_EL1 = X[t, 64];
```

A.2.5.6 DBGBVR<n>_EL1, Debug Breakpoint Value Registers, n = 0 - 5

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.

AArch64 register DBGBVR<n>_EL1 bits [63:0] are architecturally mapped to External register [B.2.2.1.13 DBGBVR<n>_EL1, Debug Breakpoint Value Registers, n = 0 - 5](#) on page 1646 bits [63:0].

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

When AArch64-DBGBCR<n>_EL1.BT == '000x'

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

When AArch64-DBGBCR<n>_EL1.BT == '001x'

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

When AArch64-DBGBCR<n>_EL1.BT == '011x'

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

When AArch64-DBGBCR<n>_EL1.BT == '100x'

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

When AArch64-DBGBCR<n>_EL1.BT == '101x'

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

When AArch64-DBGBCR<n>_EL1.BT == '110x'

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

When AArch64-DBGBCR<n>_EL1.BT == '111x'

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-DBGBCR<n>_EL1.BT == '000'

Figure A-294: AARCH64_DBGBCR<n>_EL1 bit assignments

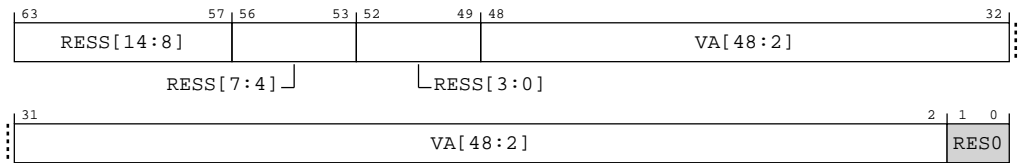


Table A-748: DBGBCR<n>_EL1 bit descriptions

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 the PE ignores this field when comparing an address, and if the breakpoint is not context-aware, the value read back in each bit of this field is a copy of the most significant bit of the VA field.	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.	47 {x}
[1:0]	RES0	Reserved	RES0

When AArch64-DBGBCR<n>_EL1.BT == '001'

Figure A-295: AARCH64_DBGBVR<n>_EL1 bit assignments

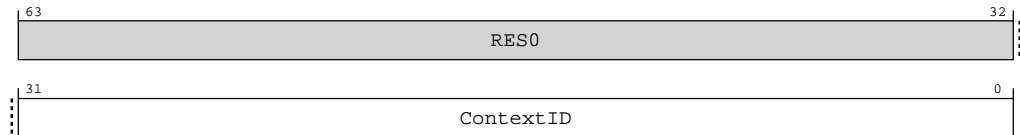


Table A-749: DBGBVR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison. The value is compared against AArch64-CONTEXTIDR_EL1.	32 {x}

When AArch64-DBGBCR<n>_EL1.BT == '011'

Figure A-296: AARCH64_DBGBVR<n>_EL1 bit assignments

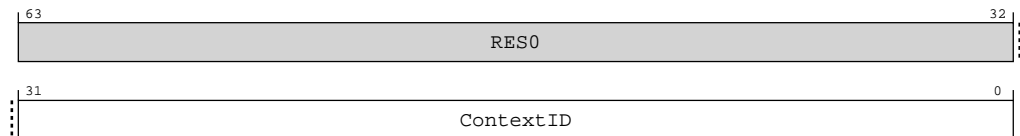


Table A-750: DBGBVR<n>_EL1 bit descriptions

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-DBGBCR<n>_EL1.BT == '100'

Figure A-297: AARCH64_DBGBVR<n>_EL1 bit assignments

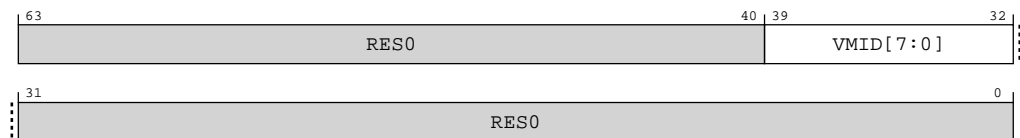
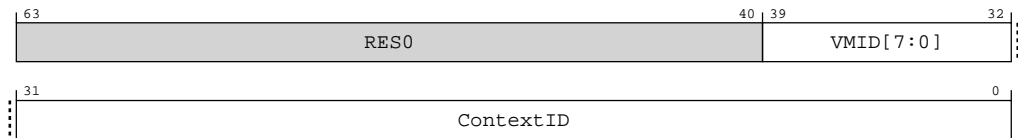


Table A-751: DBGBCR<n>_EL1 bit descriptions

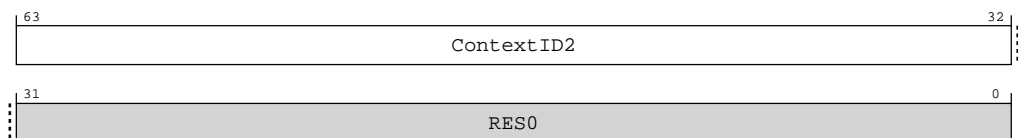
Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39:32]	VMID[7:0]	VMID value for comparison.	8 {x}
[31:0]	RES0	Reserved	RES0

When AArch64-DBGBCR<n>_EL1.BT == '101'

Figure A-298: AARCH64_DBGBCR<n>_EL1 bit assignments**Table A-752: DBGBCR<n>_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39:32]	VMID[7:0]	VMID value for comparison.	8 {x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32 {x}

When AArch64-DBGBCR<n>_EL1.BT == '110'

Figure A-299: AARCH64_DBGBCR<n>_EL1 bit assignments**Table A-753: DBGBCR<n>_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-DBGBCR<n>_EL1.BT == '111'

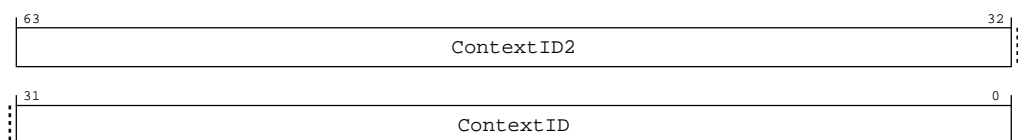
Figure A-300: AARCH64_DBGBCR<n>_EL1 bit assignments

Table A-754: DBGGBVR<n>_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>, DBGGBVR<m>_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b100

MSR DBGGBVR<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b100

Accessibility

MRS <Xt>, DBGGBVR<m>_EL1

```

integer m = UInt(CRm<3:0>);

if m >= 6 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGGBVR_EL1[m];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGGBVR_EL1[m];

```

MSR DBGGBVR<m>_EL1, <Xt>

```

integer m = UInt(CRm<3:0>);

if m >= 6 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGGBVR_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);

```

```
else
    DBGBVR_EL1[m] = X[t, 64];
```

A.2.5.7 DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 5

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.

AArch64 register DBGBCR<n>_EL1 bits [63:0] are architecturally mapped to External register [B.2.2.1.14 DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 5](#) on page 1650 bits [63:0].

Attributes

Width

64

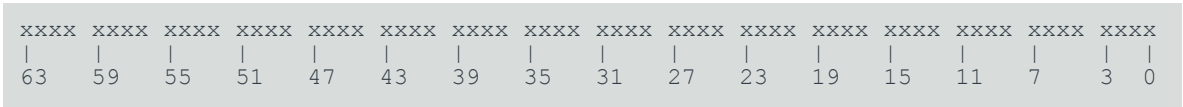
Functional group

Debug registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-301: AARCH64_DBGBCR<n>_EL1 bit assignments

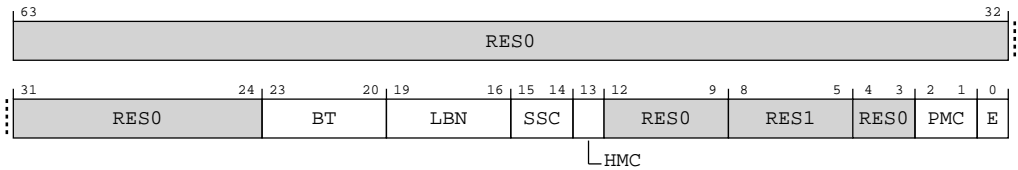


Table A-757: DBGBCR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:20]	BT	<p>Breakpoint Type.</p> <p>Specifies breakpoint type.</p> <p>0b0000 Unlinked instruction address match. AArch64-DBGBVR<n>_EL1 is the address of an instruction.</p> <p>0b0001 Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.</p> <p>0b0010 Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, the Effective value of AArch64-HCR_EL2.E2H is 1, and either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, AArch64-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b0011 As 0b0010, with linking enabled.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1000 Unlinked VMID match. AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1001 As 0b1000, with linking enabled.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1010 Unlinked VMID and Context ID match. AArch64-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and AArch64-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1011 As 0b1010, with linking enabled.</p> <p>This value applies when breakpoint n is context-aware.</p>	xxxx
[19:16]	LBN	<p>Linked Breakpoint Number.</p> <p>For Linked address matching breakpoints, specifies the index of the breakpoint linked to.</p> <p>For all other breakpoint types, this field is ignored and reads of the register return an UNKNOWN value.</p>	xxxx

Bits	Name	Description	Reset
[15:14]	SSC	<p>Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated.</p> <p>The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained.</p> <p>For more information on the operation of these fields, see <i>Execution conditions for which a breakpoint generates Breakpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>For more information on the effect of programming the fields to a reserved set of values, see <i>Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	xx
[13]	HMC	<p>Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated.</p> <p>The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained.</p> <p>For more information on the operation of these fields, see <i>Execution conditions for which a breakpoint generates Breakpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>For more information, see DBGBCR<n>_EL1.SSC.</p>	x
[12:9]	RES0	Reserved	RES0
[8:5]	RES1	Reserved	RES1
[4:3]	RES0	Reserved	RES0
[2:1]	PMC	<p>Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated.</p> <p>The fields that indicate when the breakpoint can be generated are: HMC, PMC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained.</p> <p>For more information on the operation of these fields, see <i>Execution conditions for which a breakpoint generates Breakpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>For more information, see DBGBCR<n>_EL1.SSC.</p>	xx
[0]	E	<p>Enable breakpoint n.</p> <p>0b0 Breakpoint n disabled.</p> <p>0b1 Breakpoint n enabled.</p>	x

Access

MRS <Xt>, DBGBCR<m>_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b101

MSR DBGBCR<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b101

Accessibility

MRS <Xt>, DBGBCR<m>_EL1

```
integer m = UInt(CRm<3:0>);

if m >= 6 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[m];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGBCR_EL1[m];
```

MSR DBGBCR<m>_EL1, <Xt>

```
integer m = UInt(CRm<3:0>);

if m >= 6 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGBCR_EL1[m] = X[t, 64];
```

A.2.5.8 DBGWVR<n>_EL1, Debug Watchpoint Value Registers, n = 0 - 3

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.

AArch64 register DBGWVR<n>_EL1 bits [63:0] are architecturally mapped to External register [B.2.2.1.15 DBGWVR<n>_EL1, Debug Watchpoint Value Registers, n = 0 - 3](#) on page 1653 bits [63:0].

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	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-302: AARCH64_DBGWVR<n>_EL1 bit assignments

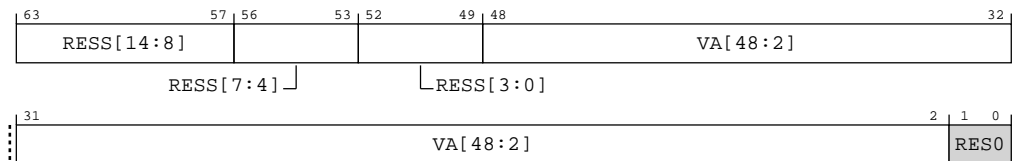


Table A-760: DBGWVR<n>_EL1 bit descriptions

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 the PE ignores this field when comparing an address, and if the breakpoint is not context-aware, the value read back in each bit of this field is a copy of the most significant bit of the VA field.	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>, DBGWVR<m>_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b110

MSR DBGWVR<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b110

Accessibility

MRS <Xt>, DBGWVR<m>_EL1

```
integer m = UInt(CRm<3:0>);

if m >= 4 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[m];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWVR_EL1[m];
```

MSR DBGWVR<m>_EL1, <Xt>

```
integer m = UInt(CRm<3:0>);

if m >= 4 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWVR_EL1[m] = X[t, 64];
```

A.2.5.9 DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 3

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.

AArch64 register DBGWCR<n>_EL1 bits [63:0] are architecturally mapped to External register [B.2.2.1.16 DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 3](#) on page 1655 bits [63:0].

Attributes

Width

64

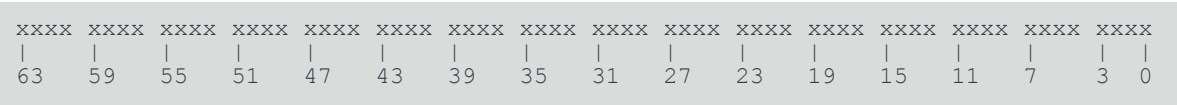
Functional group

Debug registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-303: AARCH64_DBGWCR<n>_EL1 bit assignments

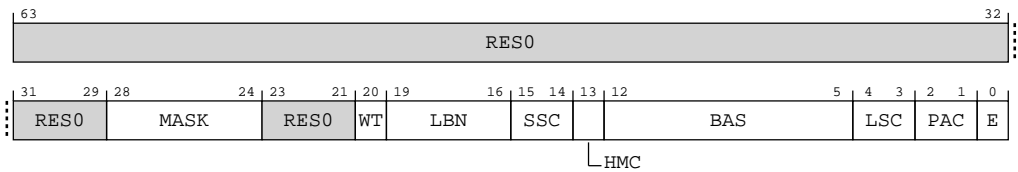


Table A-763: DBGWCR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:29]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[28:24]	MASK	<p>Address Mask. Only objects up to 2GB can be watched using a single mask.</p> <p>0b00000 No mask.</p> <p>All other values are reserved.</p> <p>Indicates the number of masked address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).</p> <p>If programmed with a reserved value, the watchpoint behaves as if either:</p> <ul style="list-style-type: none"> DBGWCR<n>_EL1.MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCR<n>_EL1. The watchpoint is disabled. 	5 {x}
[23:21]	RES0	Reserved	RES0
[20]	WT	<p>Watchpoint type. Possible values are:</p> <p>0b0 Unlinked data address match.</p> <p>0b1 Linked data address match.</p>	x
[19:16]	LBN	<p>Linked Breakpoint Number.</p> <p>For Linked data address watchpoints, specifies the index of the breakpoint linked to.</p> <p>For all other watchpoint types, this field is ignored and reads of the register return an UNKNOWN value.</p>	xxxx
[15:14]	SSC	<p>Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated.</p> <p>The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained.</p> <p>For more information on the operation of these fields, see <i>Execution conditions for which a watchpoint generates Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>For more information on the effect of programming the fields to a reserved value, see <i>Reserved DBGWCR<n>_EL1. {SSC, HMC, PAC} values</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	xx
[13]	HMC	<p>Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated.</p> <p>The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained.</p> <p>For more information on the operation of these fields, see <i>Execution conditions for which a watchpoint generates Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x

Bits	Name	Description	Reset
[12:5]	BAS	<p>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-764: BAS description on page 1051</p> <p>In cases where AArch64-DBGWVR<n>_EL1 addresses a double-word: Table A-765: BAS description table 2 on page 1051</p> <p>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.</p> <p>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 <i>Reserved DBGWCR<n>_EL1.BAS values</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	8 {x}
[4:3]	LSC	<p>Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:</p> <p>0b01 Match instructions that load from a watchpointed address.</p> <p>0b10 Match instructions that store to a watchpointed address.</p> <p>0b11 Match instructions that load from or store to a watchpointed address.</p> <p>All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.</p>	xx
[2:1]	PAC	<p>Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated.</p> <p>The fields that indicate when the watchpoint can be generated are: HMC, PAC, and SSC. These fields must be considered in combination, and the values that are permitted for these fields are constrained.</p> <p>For more information on the operation of these fields, see <i>Execution conditions for which a watchpoint generates Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	xx
[0]	E	<p>Enable watchpoint n.</p> <p>0b0 Watchpoint n disabled.</p> <p>0b1 Watchpoint n enabled.</p>	x

Table A-764: BAS description

BAS	Description
xxxxxx1	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
xxx1xxx	Match byte at AArch64-DBGWVR<n>_EL1 + 3

Table A-765: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
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>, DBGWCR<m>_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b111

MSR DBGWCR<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b111

Accessibility

MRS <Xt>, DBGWCR<m>_EL1

```
integer m = UInt(CRm<3:0>);

if m >= 4 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[m];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        X[t, 64] = DBGWCR_EL1[m];
```

MSR DBGWCR<m>_EL1, <Xt>

```
integer m = UInt(CRm<3:0>);

if m >= 4 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        DBGWCR_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() && EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
```



```
DBGWCR_EL1[m] = X[t, 64];
```

A.2.5.10 MDRAR_EL1, Monitor Debug ROM Address Register

Defines the base physical address of a 4KB-aligned memory-mapped debug component, usually a ROM table that locates and describes the memory-mapped debug components in the system. Armv8 deprecates any use of this register.

Configurations

This register is available in all configurations.

Attributes

Width

64

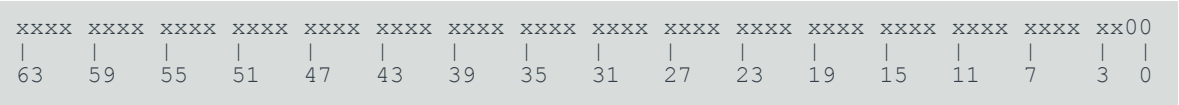
Functional group

Debug registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-304: AARCH64_MDRAR_EL1 bit assignments

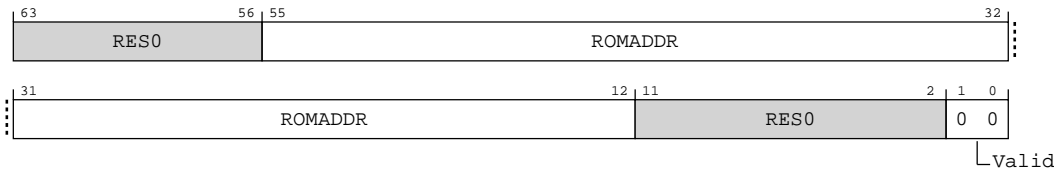


Table A-768: MDRAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0

Bits	Name	Description	Reset
55:12	ROMADDR	ROMADDR encoding for None 43:0 Reserved, UNKNOWN .	44 {x}
[11:2]	RES0	Reserved	RES0
[1:0]	Valid	This field indicates whether the ROM Table address is valid. 0b00 ROM Table address is not valid. Software must ignore ROMADDR. Other values are reserved. Arm recommends implementations set this field to zero.	0b00

Access

MRS <Xt>, MDRAR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0000	0b000

Accessibility

MRS <Xt>, MDRAR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDRA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MDRAR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MDRAR_EL1;

```

A.2.5.11 OSLAR_EL1, OS Lock Access Register

Used to lock or unlock the OS Lock.

Configurations

AArch64 register OSLAR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.1.10 OSLAR_EL1, OS Lock Access Register](#) on page 1637 bits [31:0].

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	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-305: AARCH64_OSLAR_EL1 bit assignments

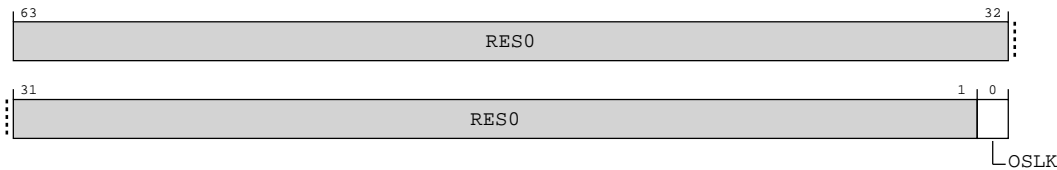


Table A-770: OSLAR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	OSLK	On writes to OSLAR_EL1, bit[0] is copied to the OS Lock. Use AArch64-OSLSR_EL1.OSLK to check the current status of the lock.	x

Access

MSR OSLAR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0000	0b100

Accessibility

MSR OSLAR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        OSLAR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    OSLAR_EL1 = X[t, 64];
```

A.2.5.12 OSLSR_EL1, OS Lock Status Register

Provides the status of the OS Lock.

Configurations

This register is available in all configurations.

Attributes

Width

64

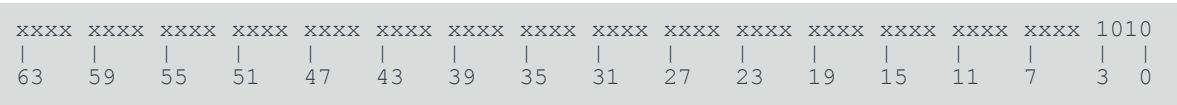
Functional group

Debug registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-306: AARCH64_OSLSR_EL1 bit assignments

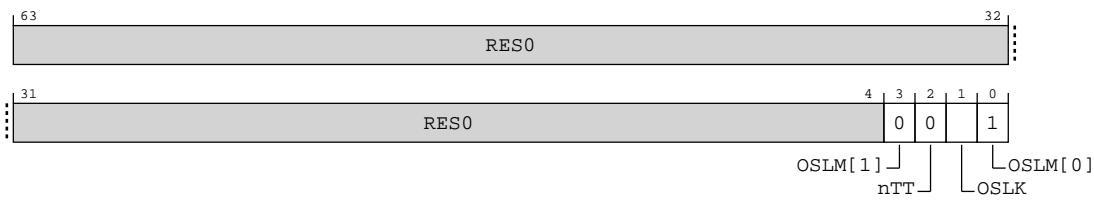


Table A-772: OSLSR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3, 0]	OSLM	OS Lock model implemented. Identifies the form of OS save and restore mechanism implemented. 0b10 OS Lock implemented.	0b10

Bits	Name	Description	Reset
[2]	nTT	Not 32-bit access. This bit is always RAZ . It indicates that a 32-bit access is needed to write the key to the OS Lock Access Register. 0b0 32-bit access.	0b0
[1]	OSLK	OS Lock Status. 0b0 OS Lock unlocked. 0b1 OS Lock locked. The OS Lock is locked and unlocked by writing to the OS Lock Access Register.	0b1

Access

MRS <Xt>, OSLSR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0001	0b100

Accessibility

MRS <Xt>, OSLSR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = OSLSR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = OSLSR_EL1;
```

A.2.5.13 OSDLR_EL1, OS Double Lock Register

Used to control the OS Double Lock.

Configurations

This register is available in all configurations.

Attributes

Width

64

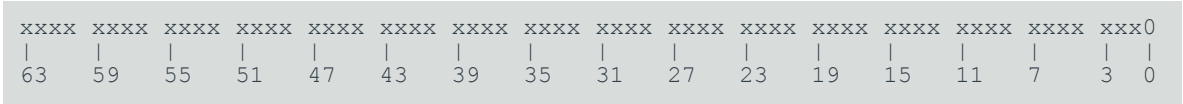
Functional group

Debug registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-307: AARCH64_OSDLR_EL1 bit assignments

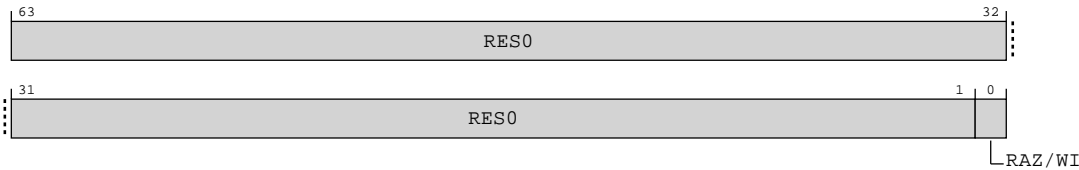


Table A-774: OSDLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	RAZ/WI	Reserved	RAZ/WI

Access

MRS <Xt>, OSDLR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0011	0b100

MSR OSDLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0011	0b100

Accessibility

MRS <Xt>, OSDLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = OSDLR_EL1;
elseif PSTATE.EL == EL2 then
```

X[t, 64] = OSDLR_EL1;

MSR OSDLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        OSDLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    OSDLR_EL1 = X[t, 64];
```

A.2.5.14 DBGPRCR_EL1, Debug Power Control Register

Controls behavior of the PE on powerdown request.

Configurations

Bit [0] of this register is mapped to ext-EDPRCR.CORENPDRQ, bit [0] of the external view of this register.

The other bits in these registers are not mapped to each other.

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	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-308: AARCH64_DBGPRCR_EL1 bit assignments

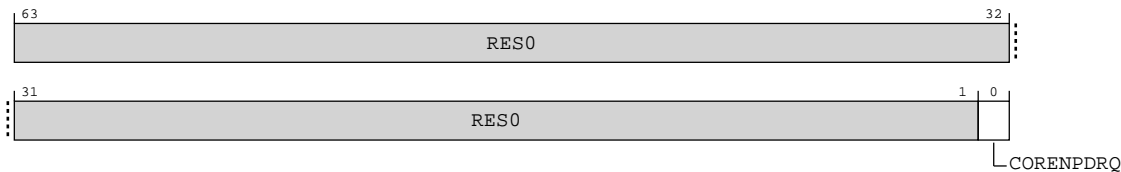


Table A-777: DBGPRCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	CORENPDRQ	<p>Core no powerdown request. Requests emulation of powerdown.</p> <p>This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1.</p> <p>0b0</p> <p>If the system responds to a powerdown request, it powers down Core power domain.</p> <p>0b1</p> <p>If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.</p> <p>This bit is not reset to the Cold reset value on exit from an IMPLEMENTATION DEFINED software-visible retention state.</p> <p>Note:</p> <p>Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.</p>	0b0

Access

MRS <Xt>, DBGPRCR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0100	0b100

MSR DBGPRCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0001	0b0100	0b100

Accessibility

MRS <Xt>, DBGPRCR_EL1

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



```

elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DBGPRCR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = DBGPRCR_EL1;

```

MSR DBGPRCR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        DBGPRCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    DBGPRCR_EL1 = X[t, 64];

```

A.2.5.15 DBGCLAIMSET_EL1, Debug CLAIM Tag Set Register

Used by software to set the CLAIM tag bits to 1.

The architecture does not define any functionality for the CLAIM tag bits.



CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the AArch64-DBGCLAIMCLR_EL1 register.

Configurations

An implementation must include eight CLAIM tag bits.

AArch64 register DBGCLAIMSET_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.1.22 DBGCLAIMSET_EL1, Debug CLAIM Tag Set Register](#) on page 1668 bits [31:0].

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

```

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

```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-309: AARCH64_DBGCLAIMSET_EL1 bit assignments

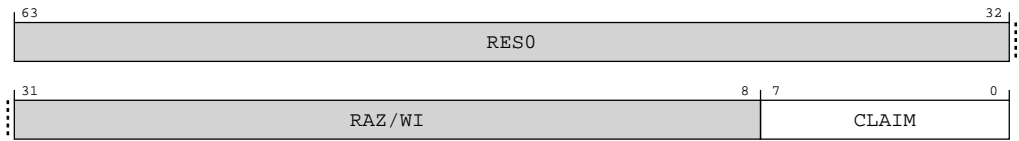


Table A-780: DBGCLAIMSET_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:8]	RAZ/WI	Reserved	RAZ/ WI
[7:0]	CLAIM	Set CLAIM tag bits. This field is RAO . Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1. Writing 0 to one of these bits has no effect.	8 {x}

Access

MRS <Xt>, DBGCLAIMSET_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1000	0b110

MSR DBGCLAIMSET_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1000	0b110

Accessibility

MRS <Xt>, DBGCLAIMSET_EL1

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

```

elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DBGCLAIMSET_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = DBGCLAIMSET_EL1;

```

MSR DBGCLAIMSET_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        DBGCLAIMSET_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    DBGCLAIMSET_EL1 = X[t, 64];

```

A.2.5.16 DBGCLAIMCLR_EL1, Debug CLAIM Tag Clear Register

Used by software to read the values of the CLAIM tag bits, and to clear CLAIM tag bits to 0.

The architecture does not define any functionality for the CLAIM tag bits.



Note

CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the AArch64-DBGCLAIMSET_EL1 register.

Configurations

An implementation must include eight CLAIM tag bits.

AArch64 register DBGCLAIMCLR_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.1.23 DBGCLAIMCLR_EL1, Debug CLAIM Tag Clear Register](#) on page 1670 bits [31:0].

Attributes

Width

64

Functional group

Debug registers

Access type

See bit descriptions

Reset value

```

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

```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-310: AARCH64_DBGCLAIMCLR_EL1 bit assignments

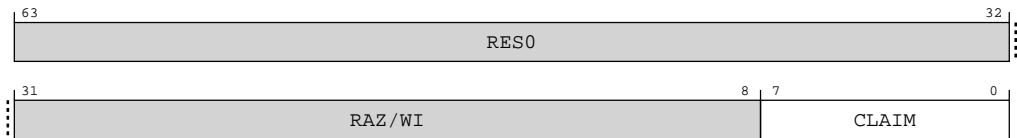


Table A-783: DBGCLAIMCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:8]	RAZ/WI	Reserved	RAZ/ WI
[7:0]	CLAIM	Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits. Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0. Writing 0 to one of these bits has no effect.	0x00

Access

MRS <Xt>, DBGCLAIMCLR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1001	0b110

MSR DBGCLAIMCLR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1001	0b110

Accessibility

MRS <Xt>, DBGCLAIMCLR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DBGCLAIMCLR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = DBGCLAIMCLR_EL1;
```

MSR DBGCLAIMCLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        DBGCLAIMCLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    DBGCLAIMCLR_EL1 = X[t, 64];
```

A.2.5.17 DBGAUTHSTATUS_EL1, Debug Authentication Status Register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for debug.

Configurations

This register is in the Core power domain.

AArch64 register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to External register [B.2.2.1.28 DBGAUTHSTATUS_EL1, Debug Authentication Status Register](#) on page 1676 bits [31:0].

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	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-311: AARCH64_DBGAUTHSTATUS_EL1 bit assignments

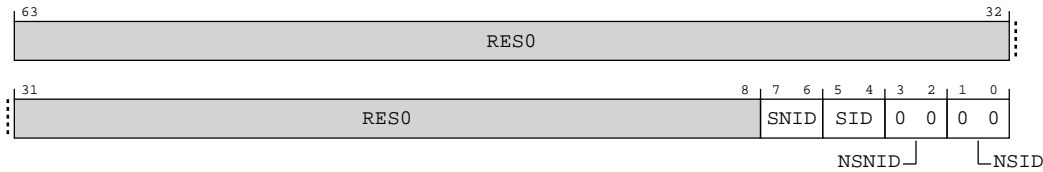


Table A-786: DBGAUTHSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure non-invasive debug. This field has the same value as DBGAUTHSTATUS_EL1.SID.	xx
[5:4]	SID	Secure invasive debug. 0b10 Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. 0b11 Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE. All other values are reserved.	xx
[3:2]	NSNID	Non-secure non-invasive debug. 0b00 Non-secure state is not implemented. All other values are reserved.	0b00
[1:0]	NSID	Non-secure invasive debug. 0b00 Non-secure state is not implemented. All other values are reserved.	0b00

Access

MRS <Xt>, DBGAUTHSTATUS_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0111	0b1110	0b110

Accessibility

MRS <Xt>, DBGAUTHSTATUS_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DBGAUTHSTATUS_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = DBGAUTHSTATUS_EL1;
```

A.2.5.18 MDCCSR_EL0, Monitor DCC Status Register

Read-only register containing control status flags for the DCC.

Configurations

AArch64 register MDCCSR_EL0 bits [30:29] are architecturally mapped to External register [B.2.2.1.6 EDSCR, External Debug Status and Control Register](#) on page 1624 bits [30:29].

Attributes

Width

64

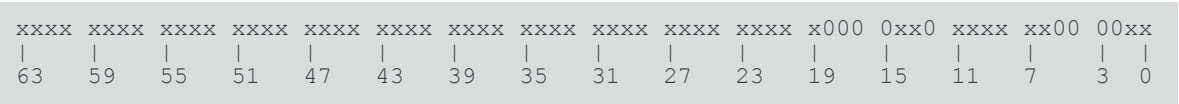
Functional group

Debug registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-312: AARCH64_MDCCSR_EL0 bit assignments

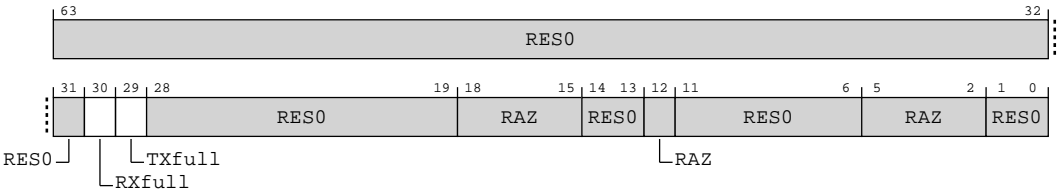


Table A-788: MDCCSR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[30]	RXfull	DTRRX full. Read-only view of the equivalent bit in the ext-EDSCR.	x
[29]	TXfull	DTRTX full. Read-only view of the equivalent bit in the ext-EDSCR.	x
[28:19]	RES0	Reserved	RES0
[18:15]	RAZ	Reserved	RAZ
[14:13]	RES0	Reserved	RES0
[12]	RAZ	Reserved	RAZ
[11:6]	RES0	Reserved	RES0
[5:2]	RAZ	Reserved	RAZ
[1:0]	RES0	Reserved	RES0

Access

MRS <Xt>, MDCCSR_ELO

op0	op1	CRn	CRm	op2
0b10	0b011	0b0000	0b0001	0b000

Accessibility

MRS <Xt>, MDCCSR_ELO

```

if Halted() then
    X[t, 64] = MDCCSR_ELO;
elsif PSTATE.EL == EL0 then
    if MDSCR_EL1.TDCC == '1' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif HCR_EL2.TGE == '1' || MDCR_EL2.<TDE,TDA> != '00' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = MDCCSR_ELO;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.<TDE,TDA> != '00' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = MDCCSR_ELO;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = MDCCSR_ELO;

```

A.2.5.19 DBGDTR_ELO, Debug Data Transfer Register, half-duplex

Transfers 64 bits of data between the PE and an external debugger. Can transfer both ways using only a single register.

Configurations

AArch64 register DBGDTR_ELO bits [63:32] are architecturally mapped to External register [B.2.2.1.4 DBGDTRRX_ELO, Debug Data Transfer Register, Receive](#) on page 1620 bits [31:0].

AArch64 register DBGDTR_ELO bits [63:32] are architecturally mapped to AArch64 System register [A.2.5.20 DBGDTRRX_ELO, Debug Data Transfer Register, Receive](#) on page 1071 bits [31:0].

AArch64 register DBGDTR_ELO bits [31:0] are architecturally mapped to External register [B.2.2.1.7 DBGDTRTX_ELO, Debug Data Transfer Register, Transmit](#) on page 1630 bits [31:0].

AArch64 register DBGDTR_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.21 DBGDTRTX_ELO, Debug Data Transfer Register, Transmit](#) on page 1073 bits [31:0].

AArch64 register DBGDTR_ELO bits [63:32] are architecturally mapped to External register [B.2.2.1.7 DBGDTRTX_ELO, Debug Data Transfer Register, Transmit](#) on page 1630 bits [31:0].

AArch64 register DBGDTR_ELO bits [63:32] are architecturally mapped to AArch64 System register [A.2.5.21 DBGDTRTX_ELO, Debug Data Transfer Register, Transmit](#) on page 1073 bits [31:0].

AArch64 register DBGDTR_ELO bits [31:0] are architecturally mapped to External register [B.2.2.1.4 DBGDTRRX_ELO, Debug Data Transfer Register, Receive](#) on page 1620 bits [31:0].

AArch64 register DBGDTR_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.20 DBGDTRRX_ELO, Debug Data Transfer Register, Receive](#) on page 1071 bits [31:0].

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	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-313: AARCH64_DBGDTR_ELO bit assignments

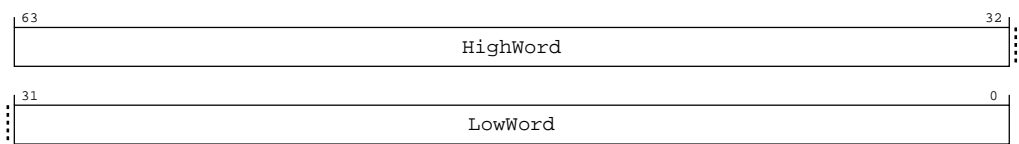


Table A-790: DBGDTR_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	HighWord	<p>Writes to this register set DTRRX to the value in this field and do not change RXfull.</p> <p>Reads of this register:</p> <ul style="list-style-type: none">If RXfull is set to 1, return the last value written to DTRTX.If RXfull is set to 0, return an UNKNOWN value. <p>After the read, RXfull is cleared to 0.</p>	32 {x}
[31:0]	LowWord	<p>Writes to this register set DTRTX to the value in this field and set TXfull to 1.</p> <p>Reads of this register:</p> <ul style="list-style-type: none">If RXfull is set to 1, return the last value written to DTRRX.If RXfull is set to 0, return an UNKNOWN value. <p>After the read, RXfull is cleared to 0.</p>	32 {x}

Access

MRS <Xt>, DBGDTR_ELO

op0	op1	CRn	CRm	op2
0b10	0b011	0b0000	0b0100	0b000

MSR DBGDTR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b0000	0b0100	0b000

Accessibility

MRS <Xt>, DBGDTR_ELO

```
if Halted() then
    X[t, 64] = DBGDTR_ELO;
elseif PSTATE.EL == EL0 then
    if MDSCR_EL1.TDCC == '1' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif HCR_EL2.TGE == '1' || MDSCR_EL2.<TDE,TDA> != '00' then
```

```

        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DBGDTR_EL0;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.<TDE,TDA> != '00' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = DBGDTR_EL0;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = DBGDTR_EL0;

```

MSR DBGDTR_EL0, <Xt>

```

if Halted() then
    DBGDTR_EL0 = X[t, 64];
elsif PSTATE.EL == EL0 then
    if MDSCR_EL1.TDCC == '1' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif HCR_EL2.TGE == '1' || MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        DBGDTR_EL0 = X[t, 64];
elsif PSTATE.EL == EL1 then
    if MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        DBGDTR_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    DBGDTR_EL0 = X[t, 64];

```

A.2.5.20 DBGDTRRX_EL0, Debug Data Transfer Register, Receive

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. See AArch64-DBGDTR_EL0 for additional architectural mappings. It is a component of the Debug Communications Channel.

Configurations

AArch64 register DBGDTRRX_EL0 bits [31:0] are architecturally mapped to External register [B.2.2.1.4 DBGDTRRX_EL0, Debug Data Transfer Register, Receive](#) on page 1620 bits [31:0].

Attributes

Width

64

Functional group

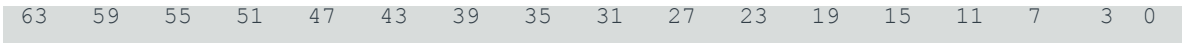
Debug registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-314: AARCH64_DBGDTRRX_ELO bit assignments

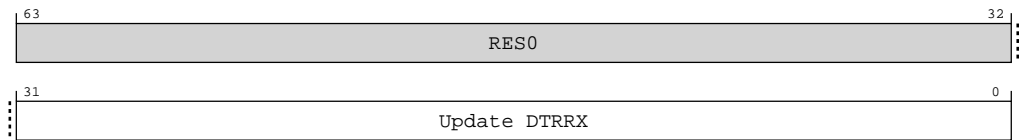


Table A-793: DBGDTRRX_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	None	Update DTRRX. Reads of this register: <ul style="list-style-type: none">If RXfull is set to 1, return the last value written to DTRRX.If RXfull is set to 0, return an UNKNOWN value. After the read, RXfull is cleared to 0. For the full behavior of the Debug Communications Channel, see <i>The Debug Communication Channel and Instruction Transfer Register</i> in the Arm® Architecture Reference Manual for A-profile architecture .	32 {x}

Access

MRS <Xt>, DBGDTRRX_ELO

op0	op1	CRn	CRm	op2
0b10	0b011	0b0000	0b0101	0b000

Accessibility

MRS <Xt>, DBGDTRRX_ELO

```
if Halted() then
    X[t, 64] = DBGDTRRX_ELO;
elseif PSTATE.EL == EL0 then
    if MDSCR_EL1.TDCC == '1' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif HCR_EL2.TGE == '1' || MDSCR_EL2.<TDE,TDA> != '00' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = DBGDTRRX_EL0;
    elsif PSTATE.EL == EL1 then
        if MDCR_EL2.<TDE,TDA> != '00' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = DBGDTRRX_EL0;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = DBGDTRRX_EL0;
```

A.2.5.21 DBGDTRTX_EL0, Debug Data Transfer Register, Transmit

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. See AArch64-DBGDTR_EL0 for additional architectural mappings. It is a component of the Debug Communication Channel.

Configurations

AArch64 register DBGDTRTX_EL0 bits [31:0] are architecturally mapped to External register [B.2.2.1.7 DBGDTRTX_EL0, Debug Data Transfer Register, Transmit](#) on page 1630 bits [31:0].

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-315: AARCH64_DBGDTRTX_EL0 bit assignments

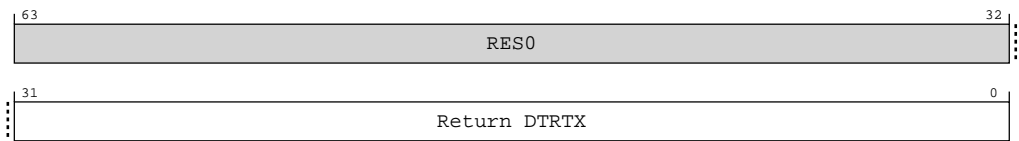


Table A-795: DBGDTRTX_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	None	<p>Return DTRTX.</p> <p>Writes to this register:</p> <ul style="list-style-type: none"> If TXfull is set to 1, set DTRRX and DTRTX to UNKNOWN. If TXfull is set to 0, update the value in DTRTX. <p>After the write, TXfull is set to 1.</p> <p>For the full behavior of the Debug Communications Channel, see <i>The Debug Communication Channel and Instruction Transfer Register</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	32 {x}

Access

MSR DBGDTRTX_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b011	0b0000	0b0101	0b000

Accessibility

MSR DBGDTRTX_ELO, <Xt>

```

if Halted() then
    DBGDTRTX_ELO = X[t, 64];
elseif PSTATE.EL == EL0 then
    if MDSCR_EL1.TDCC == '1' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif HCR_EL2.TGE == '1' || MDSCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        DBGDTRTX_ELO = X[t, 64];
elseif PSTATE.EL == EL1 then
    if MDSCR_EL2.<TDE,TDA> != '00' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        DBGDTRTX_ELO = X[t, 64];
elseif PSTATE.EL == EL2 then
    DBGDTRTX_ELO = X[t, 64];

```

A.2.5.22 TRFCR_EL1, Trace Filter Control Register (EL1)

Provides EL1 controls for Trace.

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	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-316: AARCH64_TRFCR_EL1 bit assignments

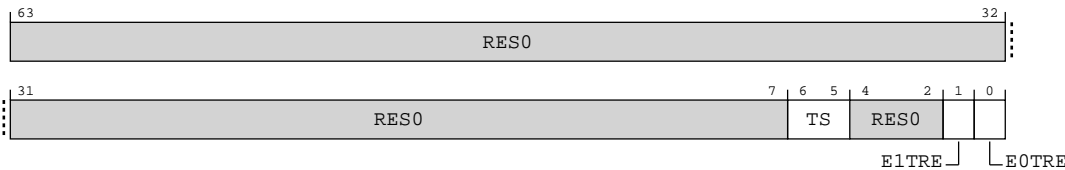


Table A-797: TRFCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:7]	RES0	Reserved	RES0
[6:5]	TS	Timestamp Control. Controls which timebase is used for trace timestamps. 0b01 Virtual timestamp. The traced timestamp is the physical counter value minus the value of AArch64-CNTVOFF_EL2. 0b11 Physical timestamp. The traced timestamp is the physical counter value. All other values are reserved. This field is ignored by the PE when any of the following are true: <ul style="list-style-type: none">EL2 is implemented and AArch64-TRFCR_EL2.TS != 0b00.SelfHostedTraceEnabled() == FALSE.	xx
[4:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1]	E1TRE	EL1 Trace Enable. 0b0 Trace is prohibited at EL1. 0b1 Trace is allowed at EL1. This field is ignored if SelfHostedTraceEnabled() == FALSE.	0b0
[0]	E0TRE	ELO Trace Enable. 0b0 Trace is prohibited at ELO. 0b1 Trace is allowed at ELO. This field is ignored if any of the following are true: <ul style="list-style-type: none">SelfHostedTraceEnabled() == FALSE.EL2 is implemented and enabled in the current Security state and AArch64-HCR_EL2.TGE == 1.	0b0

Access

MRS <Xt>, TRFCR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b001

MSR TRFCR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b001

Accessibility

MRS <Xt>, TRFCR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TTRF == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRFCR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TRFCR_EL1;
```

MSR TRFCR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if MDCR_EL2.TTRF == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```



```
TRFCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    TRFCR_EL1 = X[t, 64];
```

A.2.5.23 MDCR_EL2, Monitor Debug Configuration Register (EL2)

Provides EL2 configuration options for self-hosted debug and the Performance Monitors Extension.

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	xx0x	xxxx	xxxx	xxx0	0110
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-317: AARCH64_MDCR_EL2 bit assignments

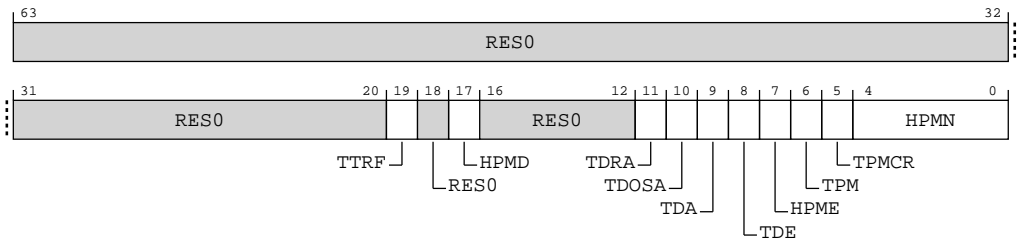


Table A-800: MDCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[19]	TTRF	<p>Traps use of the Trace Filter Control registers at EL1 to EL2, as follows:</p> <ul style="list-style-type: none"> Access to AArch64-TRFCR_EL1 is trapped to EL2, reported using EC syndrome value 0x18. <p>0b0</p> <p>Accesses to the specified registers at EL1 are not affected by this control.</p> <p>0b1</p> <p>Accesses to the specified registers at EL1 generate a trap exception to EL2 when EL2 is enabled in the current Security state.</p>	x
[18]	RES0	Reserved	RES0
[17]	HPMD	<p>Guest Performance Monitors Disable. Controls PMU operation at EL2.</p> <p>0b0</p> <p>Counters are not affected by this mechanism.</p> <p>0b1</p> <p>Affected counters are prohibited from counting at EL2.</p> <p>If AArch64-PMCR_ELO.DP is 1, then AArch64-PMCCNTR_ELO is disabled at EL2. Otherwise, AArch64-PMCCNTR_ELO is not affected by this mechanism.</p> <p>The counters affected by this field are:</p> <ul style="list-style-type: none"> Event counters AArch64-PMEVCNTR<n>_ELO for values of n less than MDCR_EL2.HPMN. If AArch64-PMCR_ELO.DP is 1, the cycle counter AArch64-PMCCNTR_ELO. <p>Other event counters are not affected by this field.</p> <p>When AArch64-PMCR_ELO.DP is 0, AArch64-PMCCNTR_ELO is not affected by this field.</p>	0b0
[16:12]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[11]	TDRA	<p>Trap Debug ROM Address register access. Traps System register accesses to the Debug ROM registers to EL2 when EL2 is enabled in the current Security state as follows:</p> <ul style="list-style-type: none"> If EL1 is using AArch64 state, accesses to AArch64-MDRAR_EL1 are trapped to EL2, reported using EC syndrome value 0x18. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>EL0 and EL1 System register accesses to the Debug ROM registers are trapped to EL2 when EL2 is enabled in the current Security state, unless it is trapped by the following:</p> <ul style="list-style-type: none"> AArch64-MDSCR_EL1.TDCC. <p>This field is treated as being 1 for all purposes other than a direct read when one or more of the following are true:</p> <ul style="list-style-type: none"> AArch64-MDCR_EL2.TDE == 1. AArch64-HCR_EL2.TGE == 1. <p>Note: EL2 does not provide traps on debug register accesses through the optional memory-mapped external debug interfaces.</p> <p>System register accesses to the debug registers might have side-effects. When a System register access is trapped to EL2, no side-effects occur before the exception is taken to EL2.</p>	x
[10]	TDOSA	<p>Trap debug OS-related register access. Traps EL1 System register accesses to the powerdown debug registers to EL2, from both Execution states as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to the following registers are trapped to EL2, reported using EC syndrome value 0x18: <ul style="list-style-type: none"> AArch64-OSLAR_EL1, AArch64-OSLSR_EL1, AArch64-OSDLR_EL1, and AArch64-DBGPRCR_EL1. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>EL1 System register accesses to the powerdown debug registers are trapped to EL2.</p> <p>Note: These registers are not accessible at EL0.</p> <p>This field is treated as being 1 for all purposes other than a direct read when one or more of the following are true:</p> <ul style="list-style-type: none"> AArch64-MDCR_EL2.TDE == 1. AArch64-HCR_EL2.TGE == 1. <p>Note: EL2 does not provide traps on debug register accesses through the optional memory-mapped external debug interfaces.</p> <p>System register accesses to the debug registers might have side-effects. When a System register access is trapped to EL2, no side-effects occur before the exception is taken to EL2.</p>	x

Bits	Name	Description	Reset
[9]	TDA	<p>Trap accesses of debug System registers. Enables a trap to EL2 on accesses of debug System registers.</p> <p>0b0</p> <p>Accesses of the specified debug System registers are not trapped by this mechanism.</p> <p>0b1</p> <p>Accesses of the specified debug System registers at EL1 and EL0 are trapped to EL2, unless the instruction generates a higher priority exception.</p> <p>In AArch64 state, the instructions affected by this control are:</p> <ul style="list-style-type: none"> MRS and MSR accesses to AArch64-DBGAUTHSTATUS_EL1, AArch64-DBGBCR<n>_EL1, AArch64-DBGBVR<n>_EL1, AArch64-DBGCLAIMCLR_EL1, AArch64-DBGCLAIMSET_EL1, AArch64-DBGWCR<n>_EL1, AArch64-DBGWVR<n>_EL1, AArch64-MDCCINT_EL1, AArch64-MDCCSR_EL0, AArch64-MDSCR_EL1, AArch64-OSDTRRX_EL1, AArch64-OSDTRTX_EL1, and AArch64-OSECCR_EL1. In Non-debug state, MRS accesses to AArch64-DBGDTRRX_EL0 and AArch64-DBGDTR_EL0 and MSR accesses to AArch64-DBGDTRTX_EL0 and AArch64-DBGDTR_EL0. <p>Unless the instruction generates a higher priority exception, trapped instructions generate an exception to EL2.</p> <p>Trapped AArch64 instructions are reported using EC syndrome value 0x18.</p> <p>The following instructions are not trapped in Debug state:</p> <ul style="list-style-type: none"> AArch64 MRS accesses to AArch64-DBGDTRRX_EL0 and AArch64-DBGDTR_EL0 and MSR accesses to AArch64-DBGDTRTX_EL0 and AArch64-DBGDTR_EL0. AArch32 MRC accesses to AArch32-DBGDTRRXint and MCR accesses to AArch32-DBGDTRTXint. <p>This field is ignored by the PE and treated as one when any of the following are true:</p> <ul style="list-style-type: none"> MDCR_EL2.TDE == 1. AArch64-HCR_EL2.TGE == 1. 	x
[8]	TDE	<p>Trap Debug Exceptions. Controls routing of Debug exceptions, and defines the debug target Exception level, EL_D.</p> <p>0b0</p> <p>The debug target Exception level is EL1.</p> <p>0b1</p> <p>If EL2 is enabled for the current Effective value of AArch64-SCR_EL3.NS, the debug target Exception level is EL2, otherwise the debug target Exception level is EL1.</p> <p>The MDCR_EL2.{TDRA, TDOSA, TDA} fields are treated as being 1 for all purposes other than returning the result of a direct read of the register.</p> <p>For more information, see <i>Routing debug exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>This field is treated as being 1 for all purposes other than a direct read when AArch64-HCR_EL2.TGE == 1.</p>	x

Bits	Name	Description	Reset
[7]	HPME	<p>Hyp Enable.</p> <p>0b0</p> <p>Affected counters are disabled and do not count.</p> <p>0b1</p> <p>Affected counters are enabled by AArch64-PMCNTENSET_ELO.</p> <p>The counters affected by this field are event counters AArch64-PMEVCNTR<n>_ELO for values of n greater than or equal to MDCR_EL2.HPMN and less than AArch64-PMCR_ELO.N.</p> <p>Other event counters and AArch64-PMCCNTR_ELO are not affected by this field.</p> <p>If MDCR_EL2.HPMN is equal to AArch64-PMCR_ELO.N, then this field has no effect.</p>	x
[6]	TPM	<p>Trap accesses of PMU registers. Enables a trap to EL2 on accesses of PMU registers.</p> <p>0b0</p> <p>Accesses of the specified PMU registers are not trapped by this mechanism.</p> <p>0b1</p> <p>Accesses of the specified PMU registers at EL1 and ELO are trapped to EL2, unless the instruction generates a higher priority exception.</p> <p>In AArch64 state, the instructions affected by this control are:</p> <ul style="list-style-type: none"> MRS and MSR accesses to AArch64-PMCCFILTR_ELO, AArch64-PMCCNTR_ELO, AArch64-PMCNTENCLR_ELO, AArch64-PMCNTENSET_ELO, AArch64-PMCR_ELO, AArch64-PMEVCNTR<n>_ELO, AArch64-PMEVTYPER<n>_ELO, AArch64-PMINTENCLR_EL1, AArch64-PMINTENSET_EL1, AArch64-PMOVSLR_ELO, AArch64-PMOVSSET_ELO, AArch64-PMSELR_ELO, AArch64-PMSWINC_ELO, AArch64-PMUSERENR_ELO, AArch64-PMXEVCNTR_ELO, and AArch64-PMXEVTYPER_ELO. MRS accesses to AArch64-PMCEIDO_ELO and AArch64-PMCEID1_ELO. If FEAT_PMUV3p4 is implemented, MRS accesses to AArch64-PMMIR_EL1. <p>Unless the instruction generates a higher priority exception, trapped instructions generate an exception to EL2.</p> <p>Trapped AArch64 instructions are reported using EC syndrome value 0x18.</p>	x
[5]	TPMCR	<p>Trap AArch64-PMCR_ELO or AArch32-PMCR accesses. Traps ELO and EL1 accesses to EL2, when EL2 is enabled in the current Security state, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to AArch64-PMCR_ELO are trapped to EL2, reported using EC syndrome value 0x18. <p>0b0</p> <p>This control does not cause any instructions to be trapped.</p> <p>0b1</p> <p>ELO and EL1 accesses to the specified registers are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by the following:</p> <ul style="list-style-type: none"> AArch64-PMUSERENR_ELO.EN. <p>Note:</p> <p>EL2 does not provide traps on Performance Monitor register accesses through the optional memory-mapped external debug interface.</p>	x

Bits	Name	Description	Reset
[4:0]	HPMN	<p>Defines the number of event counters AArch64-PMEVCNTR<n>_ELO that are accessible from EL1 and from ELO if permitted.</p> <p>MDCR_EL2.HPMN divides the event counters into a first range and a second range.</p> <p>If MDCR_EL2.HPMN is not 0 and is less than AArch64-PMCR_ELO.N, then event counters [0..(MDCR_EL2.HPMN-1)] are in the first range, and the remaining event counters [MDCR_EL2.HPMN..(AArch64-PMCR_ELO.N-1)] are in the second range.</p> <p>If MDCR_EL2.HPMN is equal to AArch64-PMCR_ELO.N, then all event counters are in the first range and none are in the second range.</p> <p>For an event counter AArch64-PMEVCNTR<n>_ELO in the first range:</p> <ul style="list-style-type: none"> The counter is accessible from EL1 and EL2. The counter is accessible from ELO if permitted by AArch64-PMUSERENR_ELO. AArch64-PMCR_ELO.E and AArch64-PMCNTENSET_ELO[n] enable the operation of the event counter. <p>For an event counter AArch64-PMEVCNTR<n>_ELO in the second range:</p> <ul style="list-style-type: none"> The counter is only accessible from EL2. MDCR_EL2.HPME and AArch64-PMCNTENSET_ELO[n] enable the operation of the event counter. <p>Values greater than AArch64-PMCR_ELO.N and 0 are reserved.</p> <p>If this field is set to a reserved value, then the following CONSTRAINED UNPREDICTABLE behaviors apply:</p> <ul style="list-style-type: none"> The value returned by a direct read of MDCR_EL2.HPMN is UNKNOWN. Either: <ul style="list-style-type: none"> An UNKNOWN number of counters are reserved for EL2 use. That is, the PE behaves as if MDCR_EL2.HPMN is set to an UNKNOWN nonzero value less than or equal to AArch64-PMCR_ELO.N. All counters are reserved for EL2 use, meaning no counters are accessible from EL1 and ELO. 	0b00110

Access

MRS <Xt>, MDCR_EL2

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

MSR MDCR_EL2, <Xt>

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

Accessibility

MRS <Xt>, MDCR_EL2

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

```
    UNDEFINED;
  elseif PSTATE.EL == EL2 then
    X[t, 64] = MDCR_EL2;
```

MSR MDCR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
  UNDEFINED;
elseif PSTATE.EL == EL1 then
  UNDEFINED;
elseif PSTATE.EL == EL2 then
  MDCR_EL2 = X[t, 64];
```

A.2.5.24 TRFCR_EL2, Trace Filter Control Register (EL2)

Provides EL2 controls for Trace.

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	x00x	0x00
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-318: AARCH64_TRFCR_EL2 bit assignments

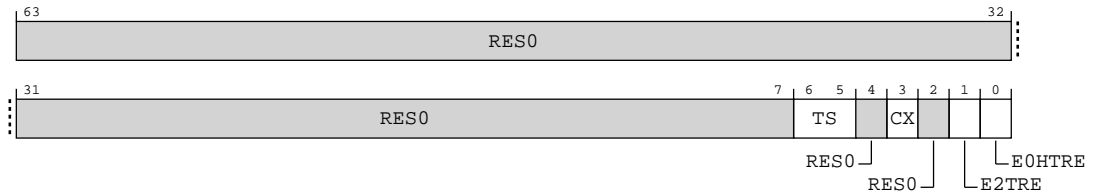


Table A-803: TRFCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:7]	RES0	Reserved	RES0
[6:5]	TS	<p>Timestamp Control. Controls which timebase is used for trace timestamps.</p> <p>0b00 Timestamp controlled by AArch64-TRFCR_EL1.TS.</p> <p>0b01 Virtual timestamp. The traced timestamp is the physical counter value minus the value of AArch64-CNTVOFF_EL2.</p> <p>0b11 Physical timestamp. The traced timestamp is the physical counter value.</p> <p>This field is ignored by the PE when <code>SelfHostedTraceEnabled() == FALSE</code>.</p>	0b00
[4]	RES0	Reserved	RES0
[3]	CX	<p>AArch64-CONTEXTIDR_EL2 and VMID trace enable.</p> <p>0b0 AArch64-CONTEXTIDR_EL2 and VMID trace prohibited.</p> <p>0b1 AArch64-CONTEXTIDR_EL2 and VMID trace allowed.</p> <p>This field is ignored if <code>SelfHostedTraceEnabled() == FALSE</code>.</p>	0b0
[2]	RES0	Reserved	RES0
[1]	E2TRE	<p>EL2 Trace Enable.</p> <p>0b0 Trace is prohibited at EL2.</p> <p>0b1 Trace is allowed at EL2.</p> <p>This field is ignored if <code>SelfHostedTraceEnabled() == FALSE</code>.</p>	0b0

Bits	Name	Description	Reset
[0]	EOHTRE	<p>ELO Trace Enable.</p> <p>0b0</p> <p>Trace is prohibited at ELO when AArch64-HCR_EL2.TGE == 1.</p> <p>0b1</p> <p>Trace is allowed at ELO when AArch64-HCR_EL2.TGE == 1.</p> <p>This field is ignored if any of the following are true:</p> <ul style="list-style-type: none"> SelfHostedTraceEnabled() == FALSE. AArch64-HCR_EL2.TGE == 0. 	0b0

Access

MRS <Xt>, TRFCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b001

MSR TRFCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b001

Accessibility

MRS <Xt>, TRFCR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = TRFCR_EL2;

```

MSR TRFCR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    TRFCR_EL2 = X[t, 64];

```

A.2.6 AArch64 GIC register description

This section includes the register descriptions for all *Generic Interrupt Controller* (GIC) registers in the Cortex®-R82 processor.

A.2.6.1 ICC_PMR_EL1, Interrupt Controller Interrupt Priority Mask Register

Provides an interrupt priority filter. Only interrupts with a higher priority than the value in this register are signaled to the PE.

Writes to this register must be high performance and must ensure that no interrupt of lower priority than the written value occurs after the write, without requiring an ISB or an exception boundary.

Configurations

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that writes to this register are self-synchronising. This ensures that no interrupts below the written PMR value will be taken after a write to this register is architecturally executed. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

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	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-319: AARCH64_ICC_PMR_EL1 bit assignments

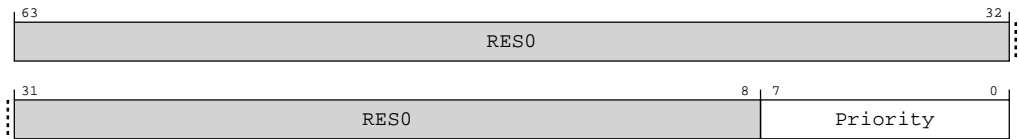


Table A-806: ICC_PMR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	Priority	<p>The priority mask level for the CPU interface. If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the PE.</p> <p>The implemented priority field values are as follows: Table A-807: Implemented priority bits description on page 1087</p> <p>Unimplemented priority bits are RAZ/WI.</p>	0x00

Table A-807: Implemented priority bits description

Implemented priority bits	Possible priority field values	Number of priority levels
[7:3]	0b00-F8 (0-248), in steps of 8	32

Access

MRS <Xt>, ICC_PMR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0110	0b000

MSR ICC_PMR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0110	0b000

Accessibility

MRS <Xt>, ICC_PMR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_PMR_EL1;
    elsif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_PMR_EL1;
    else
        X[t, 64] = ICC_PMR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ICC_PMR_EL1;

```

MSR ICC_PMR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        ICV_PMR_EL1 = X[t, 64];

```

```
elseif HCR_EL2.IMO == '1' then
    ICV_PMR_EL1 = X[t, 64];
else
    ICC_PMR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_PMR_EL1 = X[t, 64];
```

A.2.6.2 ICV_PMR_EL1, Interrupt Controller Virtual Interrupt Priority Mask Register

Provides a virtual interrupt priority filter. Only virtual interrupts with a higher priority than the value in this register are signaled to the PE.

Configurations

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that writes to this register are self-synchronising. This ensures that no interrupts below the written PMR value will be taken after a write to this register is architecturally executed. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

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	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-320: AARCH64_ICV_PMR_EL1 bit assignments

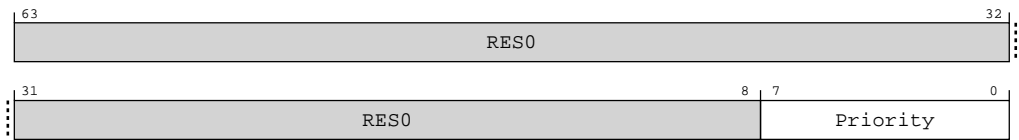


Table A-810: ICV_PMR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	Priority	<p>The priority mask level for the virtual CPU interface. If the priority of a virtual interrupt is higher than the value indicated by this field, the interface signals the virtual interrupt to the PE.</p> <p>The possible priority field values are as follows: Table A-811: Implemented priority bits description on page 1089</p> <p>Unimplemented priority bits are RAZ/WI.</p>	0x00

Table A-811: Implemented priority bits description

Implemented priority bits	Possible priority field values	Number of priority levels
[7:3]	0b00-F8 (0-248), in steps of 8	32

Access

MRS <Xt>, ICC_PMR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0110	0b000

MSR ICC_PMR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0110	0b000

Accessibility

MRS <Xt>, ICC_PMR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_PMR_EL1;
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_PMR_EL1;
    else
        X[t, 64] = ICC_PMR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_PMR_EL1;

```

MSR ICC_PMR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        ICV_PMR_EL1 = X[t, 64];
    elseif HCR_EL2.IMO == '1' then
        ICV_PMR_EL1 = X[t, 64];

```

```
else
    ICC_PMR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_PMR_EL1 = X[t, 64];
```

A.2.6.3 ICC_IAR0_EL1, Interrupt Controller Interrupt Acknowledge Register 0

The PE reads this register to obtain the INTID of the signaled Group 0 interrupt. This read acts as an acknowledge for the interrupt.

Configurations

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when PSTATE.{I,F} == {0,0}). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers'.

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	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-321: AARCH64_ICC_IAR0_EL1 bit assignments

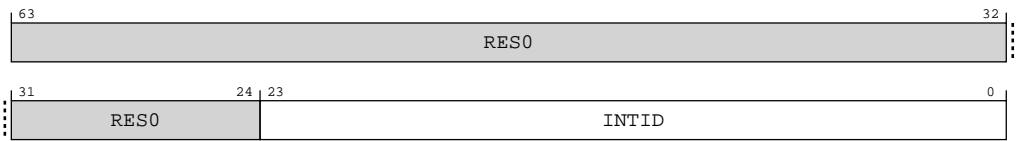


Table A-814: ICC_IAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID of the signaled interrupt.</p> <p>This is the INTID of the highest priority pending interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged at the current Security state and Exception level.</p> <p>If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).</p> <p>Bits [23:16] of this register are RES0.</p>	24 {x}

Access

MRS <Xt>, ICC_IAR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b000

Accessibility

MRS <Xt>, ICC_IAR0_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_IAR0_EL1;
    else
        X[t, 64] = ICC_IAR0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_IAR0_EL1;

```

A.2.6.4 ICV_IAR0_EL1, Interrupt Controller Virtual Interrupt Acknowledge Register 0

The PE reads this register to obtain the INTID of the signaled virtual Group 0 interrupt. This read acts as an acknowledge for the interrupt.

Configurations

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when PSTATE.{I,F} == {0,0}). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information,

see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

Width

64

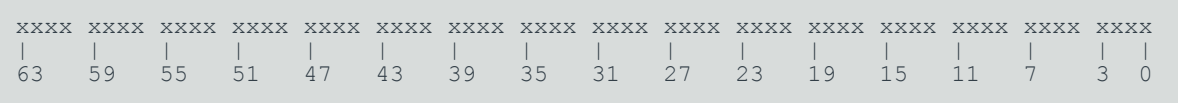
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-322: AARCH64_ICV_IAR0_EL1 bit assignments

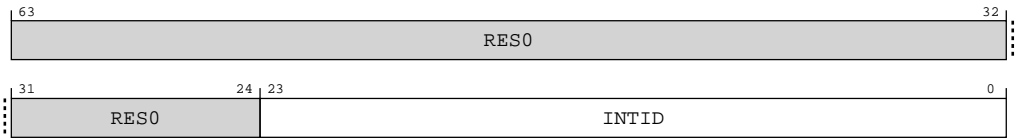


Table A-816: ICV_IAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID of the signaled virtual interrupt.</p> <p>This is the INTID of the highest priority pending virtual interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged.</p> <p>If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).</p> <p>Bits [23:16] of this register are RES0.</p>	24 {x}

Access

MRS <Xt>, ICC_IAR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b000

Accessibility

MRS <Xt>, ICC_IAR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_IAR0_EL1;
    else
        X[t, 64] = ICC_IAR0_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ICC_IAR0_EL1;
```

A.2.6.5 ICC_EOIR0_EL1, Interrupt Controller End Of Interrupt Register 0

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified Group 0 interrupt.

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	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-323: AARCH64_ICC_EOIR0_EL1 bit assignments

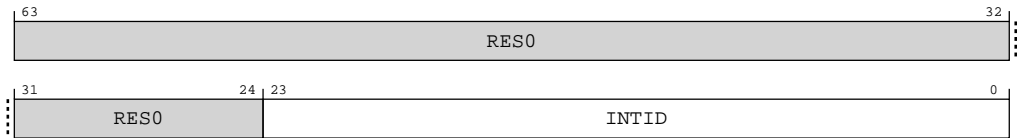


Table A-818: ICC_EOIR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID from the corresponding AArch64-ICC_IAR0_EL1 access.</p> <p>Bits [23:16] of this register are RES0.</p> <p>If AArch64-ICC_CTLR_EL1.EOIMode is 0, a write to this register drops the priority for the interrupt, and also deactivates the interrupt.</p> <p>If AArch64-ICC_CTLR_EL1.EOIMode is 1, a write to this register only drops the priority for the interrupt and software must write to AArch64-ICC_DIR_EL1 to deactivate the interrupt.</p>	24 {x}

Access

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICC_IAR0_EL1, otherwise the system behavior is **UNPREDICTABLE**. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

MSR ICC_EOIR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b001

Accessibility

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICC_IAR0_EL1, otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

MSR ICC_EOIRO_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        ICV_EOIRO_EL1 = X[t, 64];
    else
        ICC_EOIRO_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ICC_EOIRO_EL1 = X[t, 64];
```

A.2.6.6 ICV_EOIRO_EL1, Interrupt Controller Virtual End Of Interrupt Register 0

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified virtual Group 0 interrupt.

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-324: AARCH64_ICV_EOIR0_EL1 bit assignments

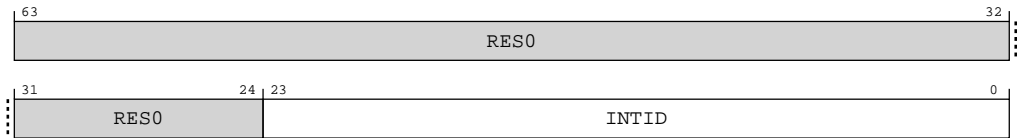


Table A-820: ICV_EOIR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID from the corresponding AArch64-ICV_IAR0_EL1 access.</p> <p>Bits [23:16] of this register are RES0.</p> <p>If the AArch64-ICV_CTLR_EL1.EOImode bit is 0, a write to this register drops the priority for the virtual interrupt, and also deactivates the virtual interrupt.</p> <p>If the AArch64-ICV_CTLR_EL1.EOImode bit is 1, a write to this register only drops the priority for the virtual interrupt. Software must write to AArch64-ICV_DIR_EL1 to deactivate the virtual interrupt.</p>	24 {x}

Access

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICV_IAR0_EL1, otherwise the system behavior is **UNPREDICTABLE**. A valid read is a read that returns a valid INTID that is not a special INTID.

MSR ICC_EOIR0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b001

Accessibility

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICV_IAR0_EL1, otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

MSR ICC_EOIR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        ICV_EOIR0_EL1 = X[t, 64];
    else
        ICC_EOIR0_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
```

```
ICC_EOIR0_EL1 = X[t, 64];
```

A.2.6.7 ICC_AP0R<n>_EL1, Interrupt Controller Active Priorities Group 0 Registers, n = 0

Provides information about Group 0 active priorities.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-325: AARCH64_ICC_AP0R<n>_EL1 bit assignments

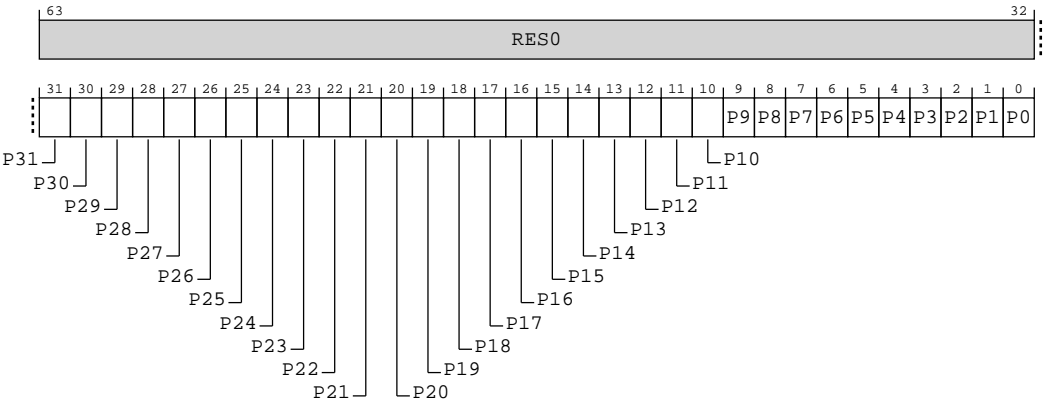


Table A-822: ICC_AP0R<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	P<x>, bit[x], where x = 31 to 0	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 priority level $x * 8$, or all active Group 0 interrupts with this priority level have undergone priority-drop. 0b1 There is a Group 0 interrupt active with priority level $x * 8$ which has not undergone priority drop.	0x00000000

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.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- AArch64-ICC_AP0R<n>_EL1.
- AArch64-ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R<m>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	'1':m[1:0]

MSR ICC_AP0R<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	'1':m[1:0]

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.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- AArch64-ICC_AP0R<n>_EL1.
- AArch64-ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP0R<m>_EL1

```
integer m = UInt(op2<1:0>);

if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_AP0R_EL1[m];
    else
        X[t, 64] = ICC_AP0R_EL1[m];
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_AP0R_EL1[m];
```

MSR ICC_AP0R<m>_EL1, <Xt>

```
integer m = UInt(op2<1:0>);

if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        ICV_AP0R_EL1[m] = X[t, 64];
    else
        ICC_AP0R_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_AP0R_EL1[m] = X[t, 64];
```

A.2.6.8 ICV_AP0R<n>_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers, n = 0

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



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-326: AARCH64_ICV_AP0R<n>_EL1 bit assignments

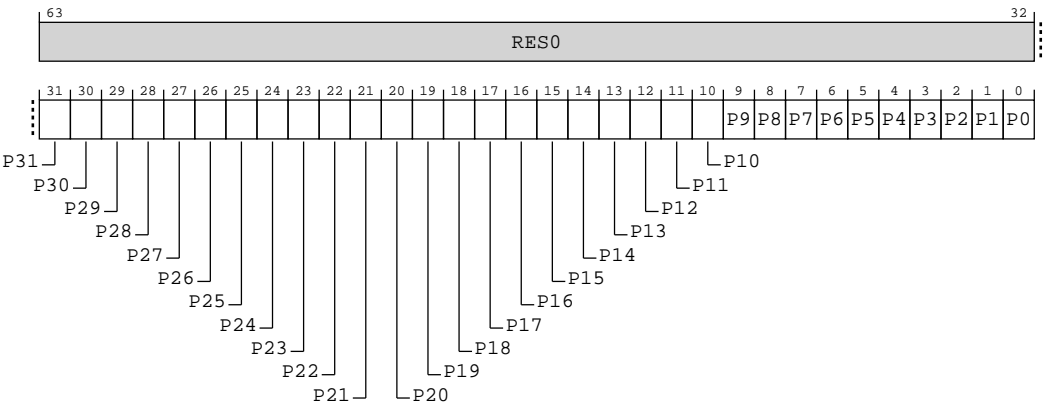


Table A-825: ICV_AP0R<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:0]	P<x>, bit[x], where x = 31 to 0	Provides the access to the active priorities for virtual Group 0 interrupts. Possible values of each bit are: 0b0 There is no virtual Group 0 interrupt active with priority level $x * 8$, or all active Group 0 interrupts with this priority level have undergone priority-drop. 0b1 There is a virtual Group 0 interrupt active with priority level $x * 8$ which has not undergone priority drop.	32 {x}

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP0R1_EL1 is 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_AP0R<m>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	'1':m[1:0]

MSR ICC_AP0R<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	'1':m[1:0]

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_AP0R<m>_EL1

```
integer m = UInt(op2<1:0>);

if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_AP0R_EL1[m];
    else
        X[t, 64] = ICC_AP0R_EL1[m];
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_AP0R_EL1[m];
```

MSR ICC_AP0R<m>_EL1, <Xt>

```
integer m = UInt(op2<1:0>);

if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        ICV_AP0R_EL1[m] = X[t, 64];
    else
        ICC_AP0R_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_AP0R_EL1[m] = X[t, 64];
```

A.2.6.9 ICC_HPPIRO_EL1, Interrupt Controller Highest Priority Pending Interrupt Register 0

Indicates the highest priority pending Group 0 interrupt on the CPU interface.

Configurations

This register is available in all configurations.

Attributes

Width

64

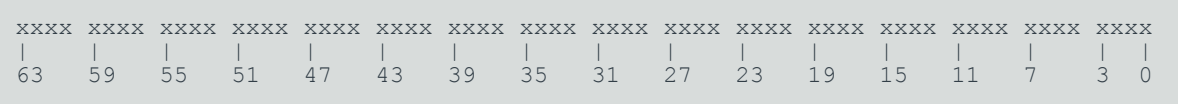
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-327: AARCH64_ICC_HPPIR0_EL1 bit assignments

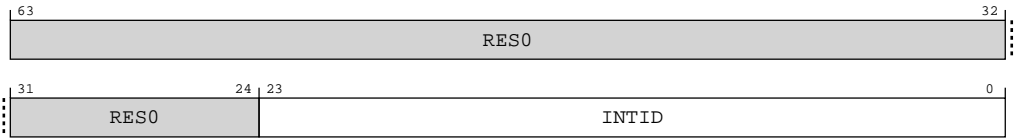


Table A-828: ICC_HPPIR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	The INTID of the highest priority pending interrupt, if that interrupt is observable at the current Security state and Exception level. If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069). Bits [23:16] of this register are RES0.	24 {x}

Access

MRS <Xt>, ICC_HPPIR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b010

Accessibility

MRS <Xt>, ICC_HPPIRO_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_HPPIRO_EL1;
    else
        X[t, 64] = ICC_HPPIRO_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ICC_HPPIRO_EL1;
```

A.2.6.10 ICV_HPPIRO_EL1, Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

Indicates the highest priority pending virtual Group 0 interrupt on the virtual CPU interface.

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-328: AARCH64_ICV_HPPIRO_EL1 bit assignments

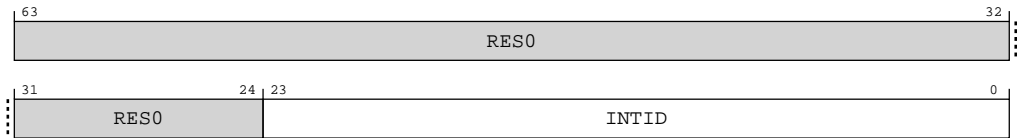


Table A-830: ICV_HPPIRO_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID of the highest priority pending virtual interrupt.</p> <p>If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).</p> <p>Bits [23:16] of this register are RES0.</p>	24 {x}

Access

MRS <Xt>, ICC_HPPIRO_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b010

Accessibility

MRS <Xt>, ICC_HPPIRO_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_HPPIRO_EL1;
    else
        X[t, 64] = ICC_HPPIRO_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_HPPIRO_EL1;
```

A.2.6.11 ICV_BPRO_EL1, Interrupt Controller Virtual Binary Point Register 0

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 0 interrupt preemption.

Configurations

This register is available in all configurations.

Attributes

Width

64

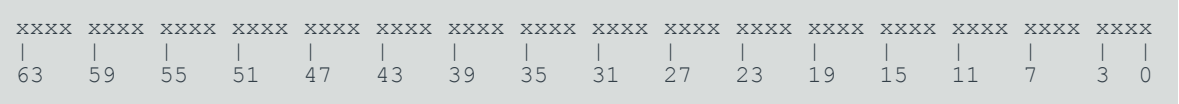
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-329: AARCH64_ICV_BPR0_EL1 bit assignments

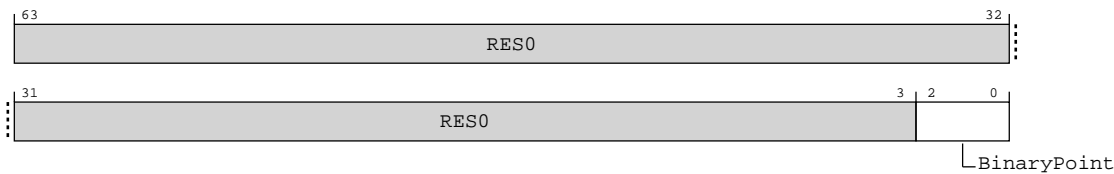


Table A-832: ICV_BPR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2:0]	BinaryPoint	The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. This is done as follows: Table A-833: Binary point value description on page 1106	xxx

Table A-833: Binary point value description

Binary point value	Group priority field	Subpriority field	Field with binary point
2	[7:3]	[2:0]	ggggg.sss
3	[7:4]	[3:0]	gggg.ssss
4	[7:5]	[4:0]	ggg.sssss
5	[7:6]	[5:0]	gg.ssssss
6	[7]	[6:0]	g.sssssss
7	No preemption	[7:0]	.ssssssss

Access

The minimum binary point value is derived from the number of implemented preemption bits, as shown in the following table:

Table A-834: Number of implemented preemption bits description

Number of implemented preemption bits	Minimum value of BPRO
7	0
6	1
5	2

The number of implemented preemption bits is indicated by AArch64-ICH_VTR_EL2.PREbits.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is **UNKNOWN**.

MRS <Xt>, ICC_BPRO_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b011

MSR ICC_BPRO_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1000	0b011

Accessibility

The minimum binary point value is derived from the number of implemented preemption bits, as shown in the following table:

+-----+-----+ Number of implemented preemption bits										Minimum value of BPRO +=====									
+=====+ 7 0 +-----																			
+-----+ 6 1 +-----+-----+ 5 2																			
+-----+-----+																			

The number of implemented preemption bits is indicated by AArch64-ICH_VTR_EL2.PREbits.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is UNKNOWN.

MRS <Xt>, ICC_BPRO_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_BPRO_EL1;
    else

```

```
        X[t, 64] = ICC_BPRO_EL1;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = ICC_BPRO_EL1;
```

MSR ICC_BPRO_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        ICV_BPRO_EL1 = X[t, 64];
    else
        ICC_BPRO_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_BPRO_EL1 = X[t, 64];
```

A.2.6.12 ICC_AP1R<n>_EL1, Interrupt Controller Active Priorities Group 1 Registers, n = 0

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	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



Where the reset reads xxxx, see individual bits.

Note

Bit descriptions

Figure A-330: AARCH64_ICC_AP1R<n>_EL1 bit assignments

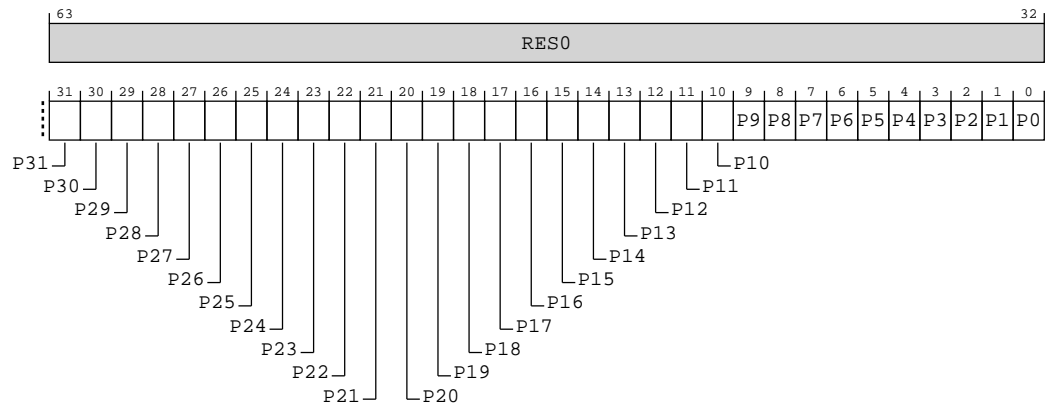


Table A-837: ICC_AP1R<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	P<x>, bit[x], where x = 31 to 0	Provides the access to the active priorities for Group 1 interrupts. Possible values of each bit are: 0b0 There is no Group 1 interrupt active with priority level x * 8, or all active Group 0 interrupts with this priority level have undergone priority-drop. 0b1 There is a Group 1 interrupt active with priority level x * 8 which has not undergone priority drop.	0x00000000

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.

Writing to the active priority registers in any order other than the following order will result in **UNPREDICTABLE** behavior:

- AArch64-ICC_AP0R<n>_EL1.
- AArch64-ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R<m>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	'0':m[1:0]

MSR ICC_AP1R<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	'0':m[1:0]

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.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- AArch64-ICC_APOR<n>_EL1.
- AArch64-ICC_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R<m>_EL1

```
integer m = UInt(op2<1:0>);

if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_AP1R_EL1[m];
    else
        X[t, 64] = ICC_AP1R_EL1[m];
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_AP1R_EL1[m];
```

MSR ICC_AP1R<m>_EL1, <Xt>

```
integer m = UInt(op2<1:0>);

if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        ICV_AP1R_EL1[m] = X[t, 64];
```

```
else
    ICC_AP1R_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_AP1R_EL1[m] = X[t, 64];
```

A.2.6.13 ICV_AP1R<n>_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers, n = 0

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-331: AARCH64_ICV_AP1R<n>_EL1 bit assignments

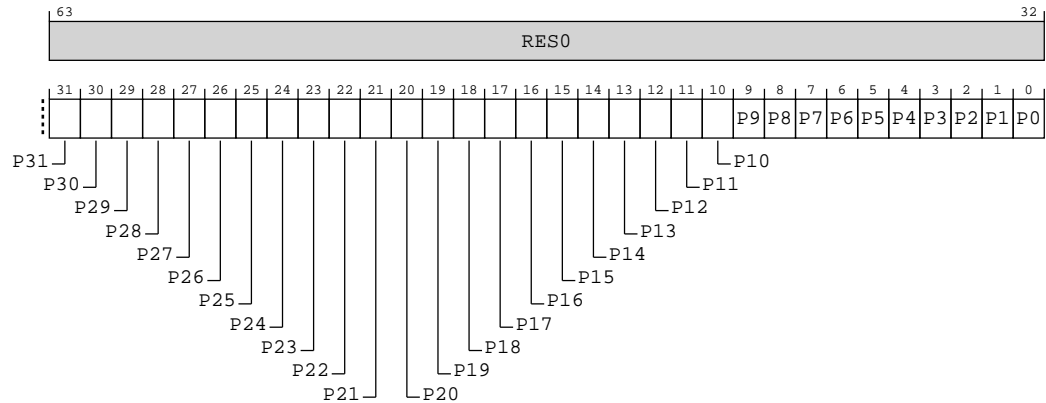


Table A-840: ICV_AP1R<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	P<x>, bit[x], where x = 31 to 0	<p>Provides the access to the active priorities for virtual Group 1 interrupts. Possible values of each bit are:</p> <p>0b0</p> <p>There is no virtual Group 1 interrupt active with priority level $x * 8$, or all active Group 1 interrupts with this priority level have undergone priority-drop.</p> <p>0b1</p> <p>There is a virtual Group 1 interrupt active with priority level $x * 8$ which has not undergone priority drop.</p>	32 {x}

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in **UNPREDICTABLE** behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP1R1_EL1 is 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_APOR<n>_EL1.

- ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R<m>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	'0':m[1:0]

MSR ICC_AP1R<m>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	'0':m[1:0]

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_APOR<n>_EL1.
- ICV_AP1R<n>_EL1.

MRS <Xt>, ICC_AP1R<m>_EL1

```
integer m = UInt(op2<1:0>);

if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_AP1R_EL1[m];
    else
        X[t, 64] = ICC_AP1R_EL1[m];
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_AP1R_EL1[m];
```

MSR ICC_AP1R<m>_EL1, <Xt>

```
integer m = UInt(op2<1:0>);
```

```
if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        ICV_AP1R_EL1[m] = X[t, 64];
    else
        ICC_AP1R_EL1[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_AP1R_EL1[m] = X[t, 64];
```

A.2.6.14 ICC_DIR_EL1, Interrupt Controller Deactivate Interrupt Register

When interrupt priority drop is separated from interrupt deactivation, a write to this register deactivates the specified interrupt.

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-332: AArch64_ICC_DIR_EL1 bit assignments

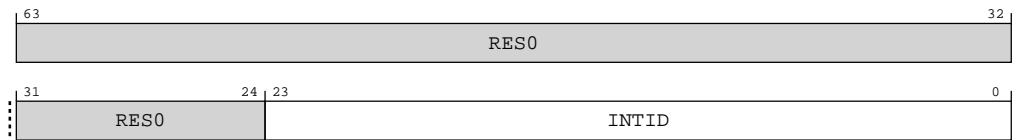


Table A-843: ICC_DIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	The INTID of the interrupt to be deactivated. Bits [23:16] of this register are RES0 .	24 {x}

Access

There are two cases when writing to AArch64-ICC_DIR_EL1 that were **UNPREDICTABLE** for a corresponding GICv2 write to ext-GICC_DIR:

- When EOImode == 0. GICv3 implementations must ignore such writes. In systems supporting system error generation, an implementation might generate an SEI.
- When EOImode == 1 but no EOI has been issued. The interrupt will be de-activated by the Distributor, however the active priority in the CPU interface for the interrupt will remain set (because no EOI was issued).

MSR ICC_DIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	0b001

Accessibility

There are two cases when writing to AArch64-ICC_DIR_EL1 that were **UNPREDICTABLE** for a corresponding GICv2 write to ext-GICC_DIR:

- When EOImode == 0. GICv3 implementations must ignore such writes. In systems supporting system error generation, an implementation might generate an SEI.
- When EOImode == 1 but no EOI has been issued. The interrupt will be de-activated by the Distributor, however the active priority in the CPU interface for the interrupt will remain set (because no EOI was issued).

MSR ICC_DIR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TDIR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        ICC_DIR_EL1 = X[t, 64];

```

```
elseif HCR_EL2.IMO == '1' then
    ICV_DIR_EL1 = X[t, 64];
else
    ICC_DIR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_DIR_EL1 = X[t, 64];
```

A.2.6.15 ICV_DIR_EL1, Interrupt Controller Deactivate Virtual Interrupt Register

When interrupt priority drop is separated from interrupt deactivation, a write to this register deactivates the specified virtual interrupt.

Configurations

This register is available in all configurations.

Attributes

Width

64

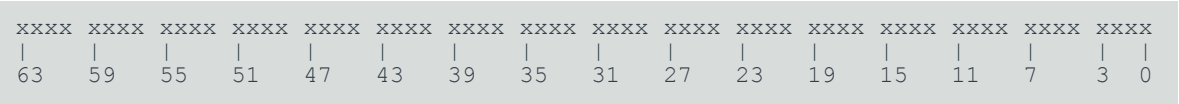
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-333: AARCH64_ICV_DIR_EL1 bit assignments

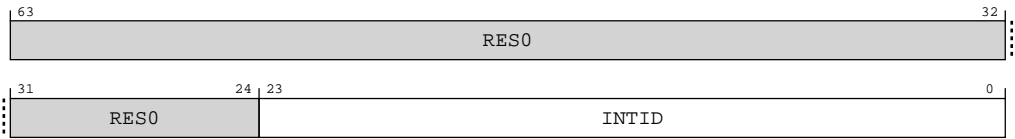


Table A-845: ICV_DIR_EL1 bit descriptions

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

Bits	Name	Description	Reset
[23:0]	INTID	The INTID of the interrupt to be deactivated. Bits [23:16] of this register are RES0 .	24{x}

Access

When EOImode == 0, writes are ignored. In systems supporting system error generation, an implementation might generate an SEI.

MSR ICC_DIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	0b001

Accessibility

When EOImode == 0, writes are ignored. In systems supporting system error generation, an implementation might generate an SEI.

MSR ICC_DIR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TDIR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        ICV_DIR_EL1 = X[t, 64];
    elseif HCR_EL2.IMO == '1' then
        ICV_DIR_EL1 = X[t, 64];
    else
        ICC_DIR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_DIR_EL1 = X[t, 64];

```

A.2.6.16 ICC_RPR_EL1, Interrupt Controller Running Priority Register

Indicates the Running priority of the CPU interface.

Configurations

This register is available in all configurations.

Attributes

Width

64

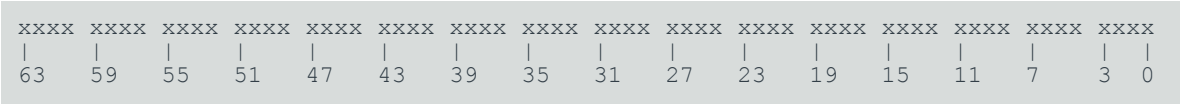
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-334: AArch64_ICC_RPR_EL1 bit assignments

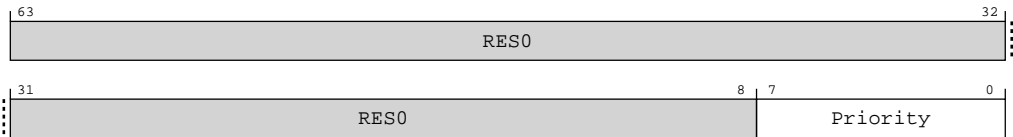


Table A-847: ICC_RPR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	Priority	<p>The current running priority on the CPU interface. This is the group priority of the current active interrupt.</p> <p>The group priority of a Secure NMI, or NMI when GICD_CTLR.DS is 1, is 0x00. The group priority of a Non-secure NMI is 0x80, saturated to 0x00 for Non-secure reads.</p> <p>If there are no active interrupts on the CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.</p> <p>The priority returned is the group priority as if the BPR for the current Exception level and Security state was set to the minimum value of BPR for the number of implemented priority bits.</p> <p>Note: If 8 bits of priority are implemented the group priority is bits[7:1] of the priority.</p>	8 {x}

Access

Software cannot determine the number of implemented priority bits from a read of this register.

MRS <Xt>, ICC_RPR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	0b011

Accessibility

Software cannot determine the number of implemented priority bits from a read of this register.

MRS <Xt>, ICC_RPR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_RPR_EL1;
    elsif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_RPR_EL1;
    else
        X[t, 64] = ICC_RPR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ICC_RPR_EL1;
```

A.2.6.17 ICV_RPR_EL1, Interrupt Controller Virtual Running Priority Register

Indicates the Running priority of the virtual CPU interface.

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-335: AARCH64_ICV_RPR_EL1 bit assignments

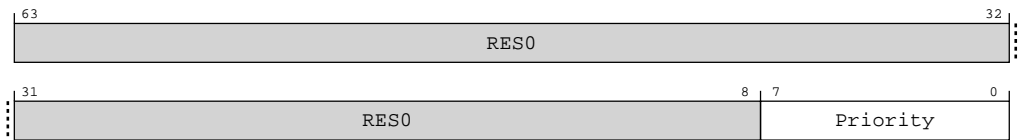


Table A-849: ICV_RPR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	Priority	<p>The current running priority on the virtual CPU interface. This is the group priority of the current active virtual interrupt.</p> <p>If there are no active interrupts on the virtual CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.</p> <p>The priority returned is the group priority as if the BPR for the current Exception level and Security state was set to the minimum value of BPR for the number of implemented priority bits.</p> <p>Note: If 8 bits of priority are implemented the group priority is bits[7:1] of the priority.</p>	8 {x}

Access

If there are no active interrupts on the virtual CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

Software cannot determine the number of implemented priority bits from a read of this register.

MRS <Xt>, ICC_RPR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	0b011

Accessibility

If there are no active interrupts on the virtual CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

Software cannot determine the number of implemented priority bits from a read of this register.

MRS <Xt>, ICC_RPR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
```

```

X[t, 64] = ICV_RPR_EL1;
elseif HCR_EL2.IMO == '1' then
    X[t, 64] = ICV_RPR_EL1;
else
    X[t, 64] = ICC_RPR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_RPR_EL1;

```

A.2.6.18 ICC_SGI1R_EL1, Interrupt Controller Software Generated Interrupt Group 1 Register

Generates Group 1 SGIs for the current Security state.

Configurations

Under certain conditions a write to ICC_SGI1R_EL1 can generate Group 0 interrupts, see 'Forwarding an SGI to a target PE' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

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-336: AARCH64_ICC_SGI1R_EL1 bit assignments

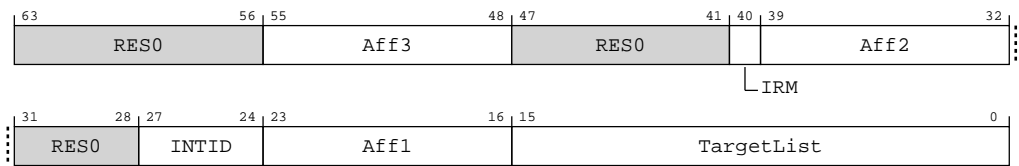


Table A-851: ICC_SGI1R_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:48]	Aff3	The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 {x}
[47:41]	RES0	Reserved	RES0
[40]	IRM	Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are: 0b0 Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>. 0b1 Interrupts routed to all PEs in the system, excluding "self".	x
[39:32]	Aff2	The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 {x}
[31:28]	RES0	Reserved	RES0
[27:24]	INTID	The INTID of the SGI.	xxxx
[23:16]	Aff1	The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 {x}
[15:0]	TargetList	Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number. If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error. If the IRM bit is 1, this field is RES0 .	16 {x}

Access

MSR ICC_SGI1R_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	0b101

Accessibility

MSR ICC_SGI1R_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.IMO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICC_SGI1R_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        ICC_SGI1R_EL1 = X[t, 64];

```

A.2.6.19 ICC_ASGI1R_EL1, Interrupt Controller Alias Software Generated Interrupt Group 1 Register

Generates Group 1 SGIs for the Security state that is not the current Security state.

Configurations

Under certain conditions a write to ICC_ASGI1R_EL1 can generate Group 0 interrupts, see 'Forwarding an SGI to a target PE' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

Width

64

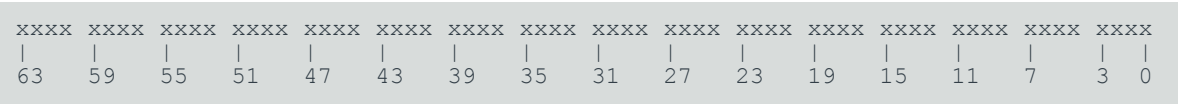
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-337: AARCH64_ICC_ASGI1R_EL1 bit assignments

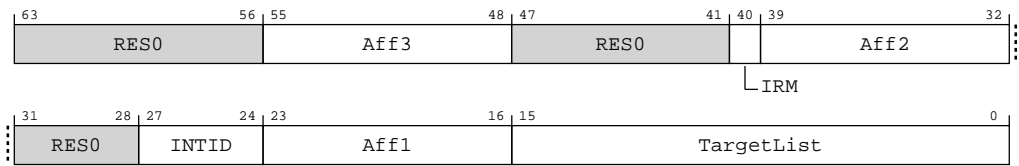


Table A-853: ICC_ASGI1R_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:48]	Aff3	The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 { x }

Bits	Name	Description	Reset
[47:41]	RES0	Reserved	RES0
[40]	IRM	Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are: 0b0 Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>. 0b1 Interrupts routed to all PEs in the system, excluding "self".	x
[39:32]	Aff2	The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 {x}
[31:28]	RES0	Reserved	RES0
[27:24]	INTID	The INTID of the SGI.	xxxx
[23:16]	Aff1	The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 {x}
[15:0]	TargetList	Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number. If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error. If the IRM bit is 1, this field is RES0 .	16 {x}

Access

This register allows software executing in a Secure state to generate Non-secure Group 1 SGIs.

MSR ICC_ASGI1R_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	0b110

Accessibility

This register allows software executing in a Secure state to generate Non-secure Group 1 SGIs.

MSR ICC_ASGI1R_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.IMO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICC_ASGI1R_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        ICC_ASGI1R_EL1 = X[t, 64];

```


A.2.6.20 ICC_SGI0R_EL1, Interrupt Controller Software Generated Interrupt Group 0 Register

Generates Secure Group 0 SGIs.

Configurations

This register is available in all configurations.

Attributes

Width

64

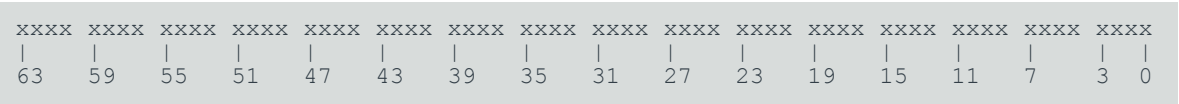
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-338: AARCH64_ICC_SGI0R_EL1 bit assignments

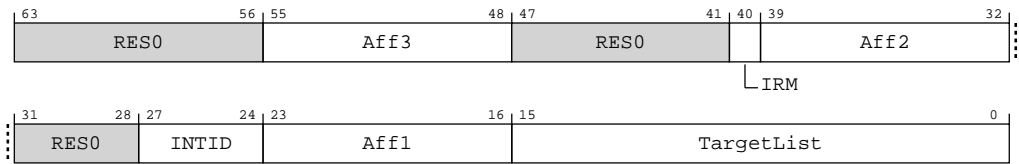


Table A-855: ICC_SGI0R_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:48]	Aff3	The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0.	8 {x}
[47:41]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[40]	IRM	Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are: 0b0 Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>. 0b1 Interrupts routed to all PEs in the system, excluding "self".	x
[39:32]	Aff2	The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 { x }
[31:28]	RES0	Reserved	RES0
[27:24]	INTID	The INTID of the SGI.	xxxx
[23:16]	Aff1	The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated. If the IRM bit is 1, this field is RES0 .	8 { x }
[15:0]	TargetList	Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number. If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error. If the IRM bit is 1, this field is RES0 .	16 { x }

Access

This register allows software executing in a Secure state to generate Group 0 SGIs. It will also allow software executing in a Non-secure state to generate Group 0 SGIs, if permitted by the settings of ext-GICR_NSACR in the Redistributor corresponding to the target PE.

When ext-GICD_CTLR.DS==0, Non-secure writes do not generate an interrupt for a target PE if not permitted by the ext-GICR_NSACR register associated with the target PE. For more information, see 'Use of control registers for SGI forwarding' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

MSR ICC_SGIOR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	0b111

Accessibility

This register allows software executing in a Secure state to generate Group 0 SGIs. It will also allow software executing in a Non-secure state to generate Group 0 SGIs, if permitted by the settings of ext-GICR_NSACR in the Redistributor corresponding to the target PE.

When ext-GICD_CTLR.DS==0, Non-secure writes do not generate an interrupt for a target PE if not permitted by the ext-GICR_NSACR register associated with the target PE. For more information, see 'Use of control registers for SGI forwarding' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

MSR ICC_SGIOR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICC_SGIOR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_SGIOR_EL1 = X[t, 64];
```

A.2.6.21 ICC_IAR1_EL1, Interrupt Controller Interrupt Acknowledge Register 1

The PE reads this register to obtain the INTID of the signaled Group 1 interrupt. This read acts as an acknowledge for the interrupt.

Configurations

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when PSTATE.{I,F} == {0,0}). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

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-339: AARCH64_ICC_IAR1_EL1 bit assignments

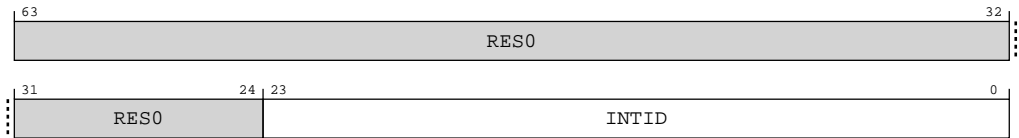


Table A-857: ICC_IAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID of the signaled interrupt.</p> <p>This is the INTID of the highest priority pending interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged at the current Security state and Exception level.</p> <p>If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).</p> <p>Bits [23:16] of this register are RES0.</p>	24 {x}

Access

MRS <Xt>, ICC_IAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b000

Accessibility

MRS <Xt>, ICC_IAR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_IAR1_EL1;
    else
        X[t, 64] = ICC_IAR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_IAR1_EL1;
```

A.2.6.22 ICV_IAR1_EL1, Interrupt Controller Virtual Interrupt Acknowledge Register 1

The PE reads this register to obtain the INTID of the signaled virtual Group 1 interrupt. This read acts as an acknowledge for the interrupt.

Configurations

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when PSTATE.{I,F} == {0,0}). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

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	0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-340: AARCH64_ICV_IAR1_EL1 bit assignments

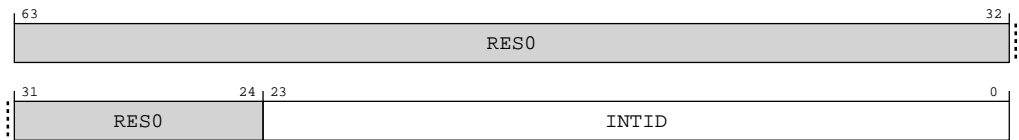


Table A-859: ICV_IAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID of the signaled virtual interrupt.</p> <p>This is the INTID of the highest priority pending virtual interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged.</p> <p>If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).</p> <p>Bits [23:16] of this register are RES0.</p>	24 {x}

Access

MRS <Xt>, ICC_IAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b000

Accessibility

MRS <Xt>, ICC_IAR1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_IAR1_EL1;
    else
        X[t, 64] = ICC_IAR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_IAR1_EL1;

```

A.2.6.23 ICC_EOIR1_EL1, Interrupt Controller End Of Interrupt Register 1

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified Group 1 interrupt.

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-341: AARCH64_ICC_EOIR1_EL1 bit assignments

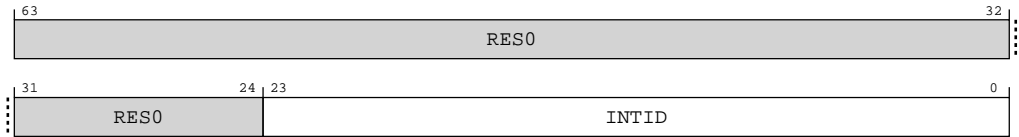


Table A-861: ICC_EOIR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	The INTID from the corresponding AArch64-ICC_IAR1_EL1 access. Bits [23:16] of this register are RES0. If AArch64-ICC_CTLR_EL1.EOIMode is 0, a write to this register drops the priority for the interrupt, and also deactivates the interrupt. If AArch64-ICC_CTLR_EL1.EOIMode is 1, a write to this register only drops the priority for the interrupt and software must write to AArch64-ICC_DIR_EL1 to deactivate the interrupt.	24 {x}

Access

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICC_IAR1_EL1, otherwise the system behavior is **UNPREDICTABLE**. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

MSR ICC_EOIR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b001

Accessibility

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICC_IAR1_EL1, otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

MSR ICC_EOIR1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.IMO == '1' then
        ICV_EOIR1_EL1 = X[t, 64];
    else
        ICC_EOIR1_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        ICC_EOIR1_EL1 = X[t, 64];

```

A.2.6.24 ICV_EOIR1_EL1, Interrupt Controller Virtual End Of Interrupt Register 1

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified virtual Group 1 interrupt.

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	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-342: AARCH64_ICV_EOIR1_EL1 bit assignments

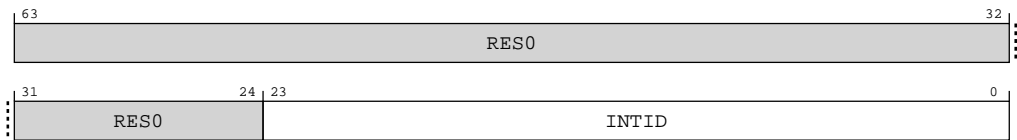


Table A-863: ICV_EOIR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	The INTID from the corresponding AArch64-ICV_IAR1_EL1 access. Bits [23:16] of this register are RES0 . If the AArch64-ICV_CTLR_EL1.EOImode bit is 0, a write to this register drops the priority for the virtual interrupt, and also deactivates the virtual interrupt. If the AArch64-ICV_CTLR_EL1.EOImode bit is 1, a write to this register only drops the priority for the virtual interrupt. Software must write to AArch64-ICV_DIR_EL1 to deactivate the virtual interrupt.	24 {x}

Access

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICV_IAR1_EL1, otherwise the system behavior is **UNPREDICTABLE**. A valid read is a read that returns a valid INTID that is not a special INTID.

MSR ICC_EOIR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b001

Accessibility

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from AArch64-ICV_IAR1_EL1, otherwise the system behavior is **UNPREDICTABLE**. A valid read is a read that returns a valid INTID that is not a special INTID.

MSR ICC_EOIR1_EL1, <Xt>

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

```
elseif PSTATE.EL == EL1 then
  if ICH_HCR_EL2.TALL1 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  elseif HCR_EL2.IMO == '1' then
    ICV_EOIR1_EL1 = X[t, 64];
  else
    ICC_EOIR1_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
  ICC_EOIR1_EL1 = X[t, 64];
```

A.2.6.25 ICC_HPPIR1_EL1, Interrupt Controller Highest Priority Pending Interrupt Register 1

Indicates the highest priority pending Group 1 interrupt on the CPU interface.

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	0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-343: AARCH64_ICC_HPPIR1_EL1 bit assignments

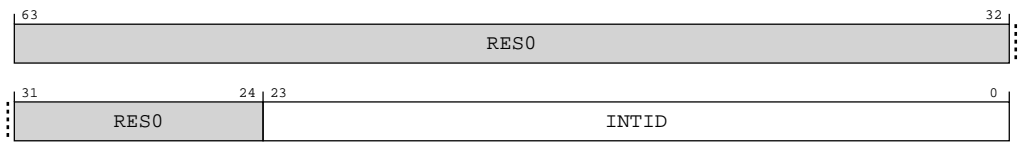


Table A-865: ICC_HPPIR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	<p>The INTID of the highest priority pending interrupt, if that interrupt is observable at the current Security state and Exception level.</p> <p>If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).</p> <p>Bits [23:16] of this register are RES0.</p>	24 {x}

Access

MRS <Xt>, ICC_HPPIR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b010

Accessibility

MRS <Xt>, ICC_HPPIR1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_HPPIR1_EL1;
    else
        X[t, 64] = ICC_HPPIR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_HPPIR1_EL1;

```

A.2.6.26 ICV_HPPIR1_EL1, Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1

Indicates the highest priority pending virtual Group 1 interrupt on the virtual CPU interface.

Configurations

This register is available in all configurations.

Attributes

Width

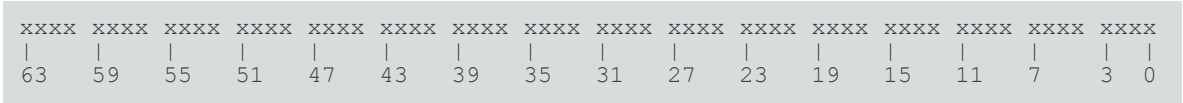
64

Functional group

GIC system registers

Access type
See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-344: AARCH64_ICV_HPPIR1_EL1 bit assignments

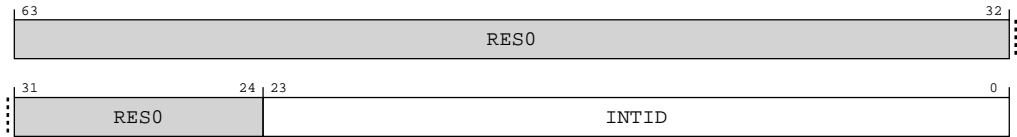


Table A-867: ICV_HPPIR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:0]	INTID	The INTID of the highest priority pending virtual interrupt. If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069). Bits [23:16] of this register are RES0.	24 {x}

Access
MRS <Xt>, ICC_HPPIR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b010

Accessibility
MRS <Xt>, ICC_HPPIR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
elseif HCR_EL2.IMO == '1' then
    X[t, 64] = ICV_HPPIR1_EL1;
else
    X[t, 64] = ICC_HPPIR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_HPPIR1_EL1;
```

A.2.6.27 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

Configurations

Virtual accesses to this register update AArch64-ICH_VMCR_EL2.VBPR1.

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 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-345: AARCH64_ICC_BPR1_EL1 bit assignments

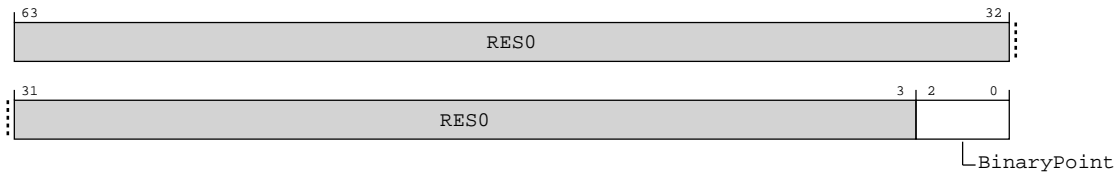


Table A-869: ICC_BPR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2:0]	BinaryPoint	If the GIC is configured to use separate binary point fields for Group 0 and Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. For more information about priorities, see 'Priority grouping' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069). The minimum value of the Secure copy of this register is the minimum value of AArch64-ICC_BPR0_EL1.	xxx

Access

On a reset, the binary point field is **UNKNOWN**.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.

MRS <Xt>, ICC_BPR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b011

MSR ICC_BPR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b011

Accessibility

On a reset, the binary point field is **UNKNOWN**.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.

MRS <Xt>, ICC_BPR1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICC_BPR1_EL1;
    else
        X[t, 64] = ICC_BPR1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_BPR1_EL1;

```

MSR ICC_BPR1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        ICC_BPR1_EL1 = X[t, 64];

```

```
else
    ICC_BPR1_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_BPR1_EL1 = X[t, 64];
```

A.2.6.28 ICV_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 1 interrupt preemption.

Configurations

This register is available in all configurations.

Attributes

Width

64

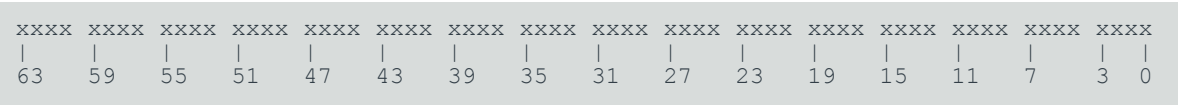
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-346: AARCH64_ICV_BPR1_EL1 bit assignments

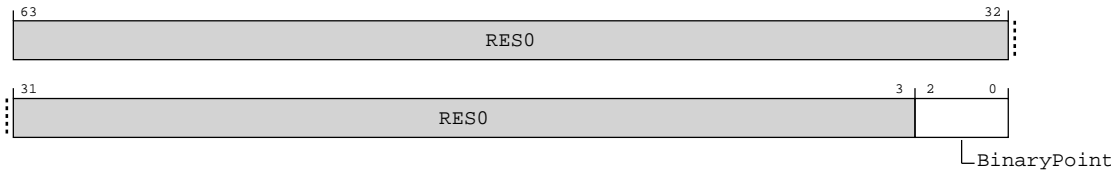


Table A-872: ICV_BPR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2:0]	BinaryPoint	<p>If the GIC is configured to use separate binary point fields for Group 0 and Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field.</p> <p>For more information about priorities, see 'Priority grouping' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).</p> <p>An attempt to program this field to a value less than the minimum value sets the field to the minimum value.</p> <p>If AArch64-ICV_CTLR_EL1.CBPR is set to 1, Secure EL1 reads return AArch64-ICV_BPR0_EL1. Secure EL1 writes modify AArch64-ICV_BPR0_EL1.</p>	xxx

Access

For Secure writes, the minimum value of this field is the minimum value of AArch64-ICH_VMCR_EL2.VBPR0.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is **UNKNOWN**.

MRS <Xt>, ICC_BPR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b011

MSR ICC_BPR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b011

Accessibility

For Secure writes, the minimum value of this field is the minimum value of AArch64-ICH_VMCR_EL2.VBPR0.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is UNKNOWN.

MRS <Xt>, ICC_BPR1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_BPR1_EL1;
    else
        X[t, 64] = ICC_BPR1_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ICC_BPR1_EL1;

```


MSR ICC_BPR1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.IMO == '1' then
        ICV_BPR1_EL1 = X[t, 64];
    else
        ICC_BPR1_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ICC_BPR1_EL1 = X[t, 64];
```

A.2.6.29 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	10xx	1000	0100	x0xx	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-347: AARCH64_ICC_CTLR_EL1 bit assignments

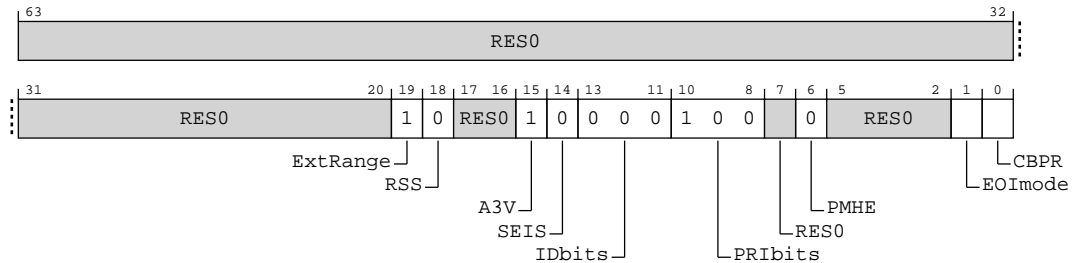


Table A-875: ICC_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0
[19]	ExtRange	Extended INTID range (read-only). 0b1 CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> All INTIDs in the range 1024..8191 are treated as requiring deactivation. 	0b1
[18]	RSS	Range Selector Support. Possible values are: 0b0 Targeted SGLs with affinity level 0 values of 0 - 15 are supported. This bit is read-only.	0b0
[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 SGL generation System registers.	0b1
[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.	0b0
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported: 0b000 16 bits. All other values are reserved.	0b000
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. For physical accesses, this field determines the minimum value of AArch64-ICC_BPR0_EL1. Physical accesses return the value from this field. 0b100 5 bits of priority are implemented	0b100
[7]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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. Read-only and writes are ignored.	0b0
[5:2]	RES0	Reserved	RES0
[1]	EOImode	EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt: 0b0 AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE . 0b1 AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.	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_BPRO_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_BPRO_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 ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    else
        X[t, 64] = ICC_CTLR_EL1;

```

```
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_CTLR_EL1;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elseif HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    else
        ICC_CTLR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_CTLR_EL1 = X[t, 64];
```

A.2.6.30 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	10xx	1000	0100	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-348: AARCH64_ICV_CTLR_EL1 bit assignments

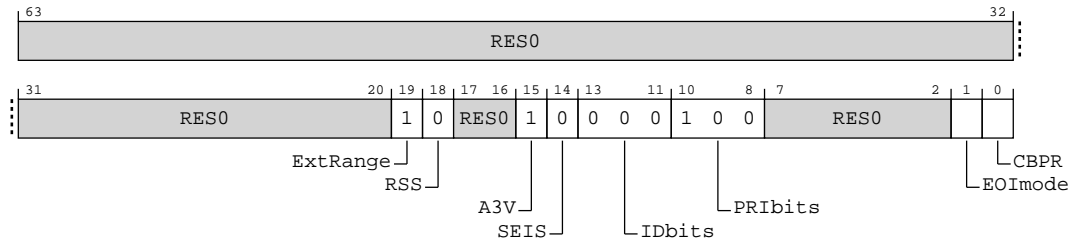


Table A-878: ICV_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0
[19]	ExtRange	Extended INTID range (read-only). 0b1 CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> All INTIDs in the range 1024..8191 are treated as requiring deactivation. ICV_CTLR_EL1.ExtRange is an alias of AArch64-ICC_CTLR_EL1.ExtRange.	0b1
[18]	RSS	Range Selector Support. Possible values are: 0b0 Targeted SGLs with affinity level 0 values of 0 - 15 are supported. This bit is read-only.	0b0
[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.	0b1
[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.	0b0
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported: 0b000 16 bits. All other values are reserved.	0b000

Bits	Name	Description	Reset
[10:8]	PRIbits	<p>Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.</p> <p>An implementation must implement at least 32 levels of physical priority (5 priority bits).</p> <p>Note: This field always returns the number of priority bits implemented.</p> <p>The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPR0_EL1 and AArch64-ICV_BPR1_EL1.</p> <p>0b100 5 bits of priority are implemented</p>	0b100
[7:2]	RES0	Reserved	RES0
[1]	EOImode	<p>Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:</p> <p>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.</p> <p>0b1 AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.</p>	x
[0]	CBPR	<p>Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts:</p> <p>0b0 AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.</p> <p>0b1 Secure reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1. Secure writes of AArch64-ICV_BPR1_EL1 modify AArch64-ICV_BPR0_EL1.</p>	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 ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then

```

```
        X[t, 64] = ICV_CTLR_EL1;
    elsif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    else
        X[t, 64] = ICC_CTLR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ICC_CTLR_EL1;
```

MSR ICC_CTLR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    else
        ICC_CTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        ICC_CTLR_EL1 = X[t, 64];
```

A.2.6.31 ICC_SRE_EL1, Interrupt Controller System Register Enable Register (EL1)

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL1.

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	x001
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-349: AARCH64_ICC_SRE_EL1 bit assignments

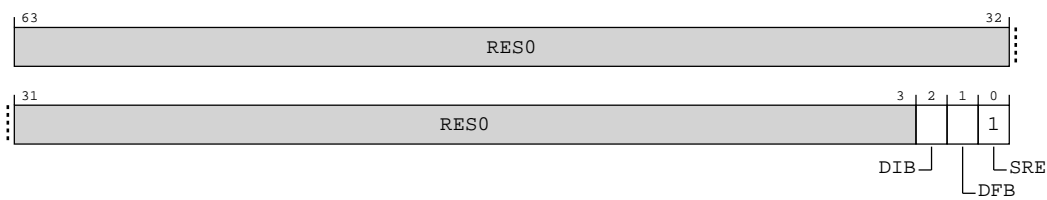


Table A-881: ICC_SRE_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2]	DIB	Disable IRQ bypass. 0b0 IRQ bypass enabled. 0b1 IRQ bypass disabled. This field is a read-only alias of AArch64-ICC_SRE_EL2.DIB.	0b0
[1]	DFB	Disable FIQ bypass. 0b0 FIQ bypass enabled. 0b1 FIQ bypass disabled. This field is a read-only alias of AArch64-ICC_SRE_EL2.DFB.	0b0
[0]	SRE	System Register Enable. 0b1 The System register interface for the current Security state is enabled.	0b1

Access

Execution with AArch64-ICC_SRE_EL1.SRE set to 0 might make some System registers **UNKNOWN**.

MRS <Xt>, ICC_SRE_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

MSR ICC_SRE_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

Accessibility

Execution with AArch64-ICC_SRE_EL1.SRE set to 0 might make some System registers UNKNOWN.

MRS <Xt>, ICC_SRE_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    X[t, 64] = ICC_SRE_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_SRE_EL1;
```

MSR ICC_SRE_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    ICC_SRE_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_SRE_EL1 = X[t, 64];
```

A.2.6.32 ICC_IGRPEN0_EL1, Interrupt Controller Interrupt Group 0 Enable register

Controls whether Group 0 interrupts are enabled or not.

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	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-350: AARCH64_ICC_IGRPEN0_EL1 bit assignments

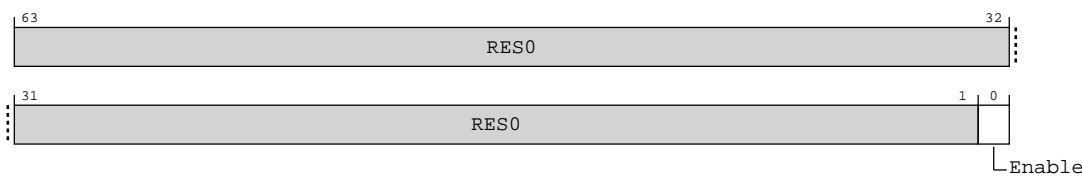


Table A-884: ICC_IGRPEN0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	Enable	Enables Group 0 interrupts. 0b0 Group 0 interrupts are disabled. 0b1 Group 0 interrupts are enabled. Virtual accesses to this register update AArch64-ICH_VMCR_EL2.VENG0. If the highest priority pending interrupt for that PE is a Group 0 interrupt using 1 of N model, then the interrupt will be targeted to another PE as a result of the Enable bit changing from 1 to 0.	0b0

Access

MRS <Xt>, ICC_IGRPEN0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

MSR ICC_IGRPEN0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

Accessibility

MRS <Xt>, ICC_IGRPEN0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_IGRPEN0_EL1;
    else
        X[t, 64] = ICC_IGRPEN0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_IGRPEN0_EL1;
```

MSR ICC_IGRPEN0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        ICV_IGRPEN0_EL1 = X[t, 64];
    else
        ICC_IGRPEN0_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ICC_IGRPEN0_EL1 = X[t, 64];
```

A.2.6.33 ICV_IGRPEN0_EL1, Interrupt Controller Virtual Interrupt Group 0 Enable register

Controls whether virtual Group 0 interrupts are enabled or not.

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-351: AARCH64_ICV_IGRPEN0_EL1 bit assignments

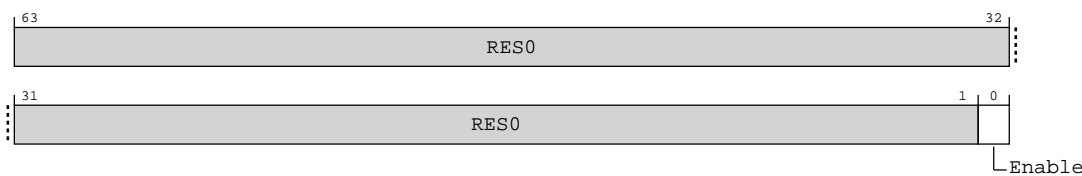


Table A-887: ICV_IGRPEN0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	Enable	Enables virtual Group 0 interrupts. 0b0 Virtual Group 0 interrupts are disabled. 0b1 Virtual Group 0 interrupts are enabled.	x

Access

MRS <Xt>, ICC_IGRPEN0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

MSR ICC_IGRPEN0_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b110

Accessibility

MRS <Xt>, ICC_IGRPEN0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_IGRPEN0_EL1;
    else
        X[t, 64] = ICC_IGRPEN0_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_IGRPEN0_EL1;
```

MSR ICC_IGRPEN0_EL1, <Xt>

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

```
elseif PSTATE.EL == EL1 then
  if ICH_HCR_EL2.TALL0 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  elseif HCR_EL2.FMO == '1' then
    ICV_IGRPEN0_EL1 = X[t, 64];
  else
    ICC_IGRPEN0_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
  ICC_IGRPEN0_EL1 = X[t, 64];
```

A.2.6.34 ICC_IGRPEN1_EL1, Interrupt Controller Interrupt Group 1 Enable register

Controls whether Group 1 interrupts are enabled for the current Security state.

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	xxx0
63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-352: AARCH64_ICC_IGRPEN1_EL1 bit assignments

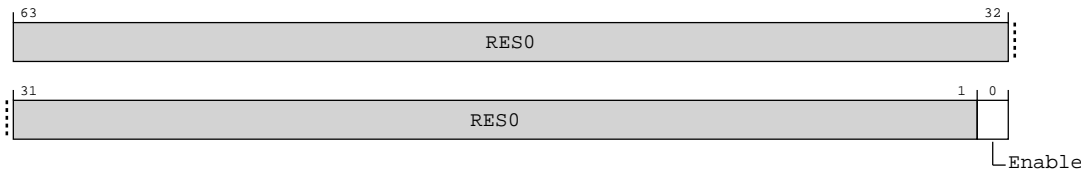


Table A-890: ICC_IGRPEN1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	Enable	<p>Enables Group 1 interrupts for the current Security state.</p> <p>0b0</p> <p>Group 1 interrupts are disabled for the current Security state.</p> <p>0b1</p> <p>Group 1 interrupts are enabled for the current Security state.</p> <p>Virtual accesses to this register update AArch64-ICH_VMCR_EL2.VENG1.</p> <p>If the highest priority pending interrupt for that PE is a Group 1 interrupt using 1 of N model, then the interrupt will target another PE as a result of the Enable bit changing from 1 to 0.</p>	0b0

Access

MRS <Xt>, ICC_IGRPEN1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b111

MSR ICC_IGRPEN1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b111

Accessibility

MRS <Xt>, ICC_IGRPEN1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_IGRPEN1_EL1;
    else
        X[t, 64] = ICC_IGRPEN1_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICC_IGRPEN1_EL1;

```

MSR ICC_IGRPEN1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HCR_EL2.IMO == '1' then
        ICV_IGRPEN1_EL1 = X[t, 64];
    else
        ICC_IGRPEN1_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then

```

```
ICC_IGRPEN1_EL1 = X[t, 64];
```

A.2.6.35 ICV_IGRPEN1_EL1, Interrupt Controller Virtual Interrupt Group 1 Enable register

Controls whether virtual Group 1 interrupts are enabled for the current Security state.

Configurations

This register is available in all configurations.

Attributes

Width

64

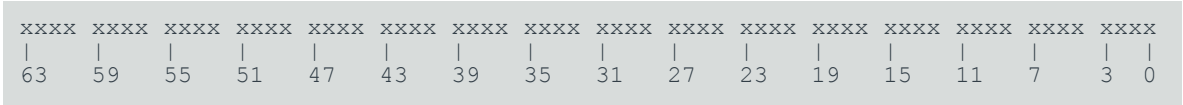
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-353: AARCH64_ICV_IGRPEN1_EL1 bit assignments

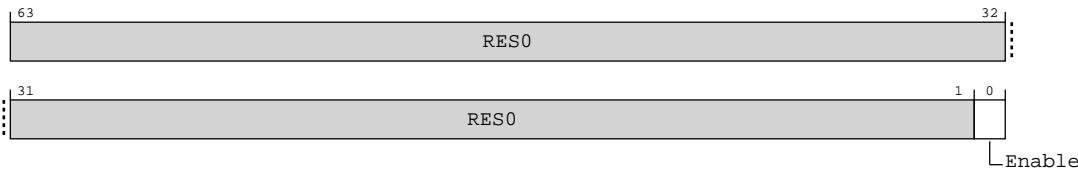


Table A-893: ICV_IGRPEN1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	Enable	Enables virtual Group 1 interrupts. 0b0 Virtual Group 1 interrupts are disabled. 0b1 Virtual Group 1 interrupts are enabled.	x

Access

MRS <Xt>, ICC_IGRPEN1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b111

MSR ICC_IGRPEN1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b111

Accessibility

MRS <Xt>, ICC_IGRPEN1_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_IGRPEN1_EL1;
    else
        X[t, 64] = ICC_IGRPEN1_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ICC_IGRPEN1_EL1;

```

MSR ICC_IGRPEN1_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if ICH_HCR_EL2.TALL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HCR_EL2.IMO == '1' then
        ICV_IGRPEN1_EL1 = X[t, 64];
    else
        ICC_IGRPEN1_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        ICC_IGRPEN1_EL1 = X[t, 64];

```


A.2.6.36 ICH_LR<n>_EL2, Interrupt Controller List Registers, n = 0 - 15

Provides interrupt context information for the virtual CPU interface.

Configurations

If list register n is not implemented, then accesses to this register are UNDEFINED.

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	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-354: AARCH64_ICH_LR<n>_EL2 bit assignments

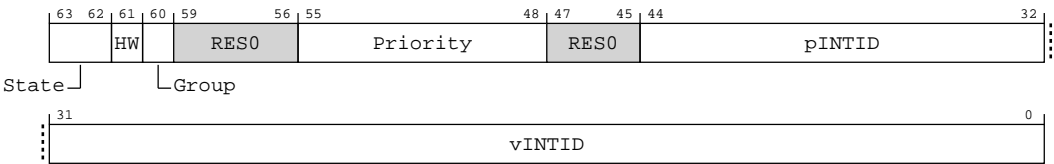


Table A-896: ICH_LR<n>_EL2 bit descriptions

Bits	Name	Description	Reset
[63:62]	State	<p>The state of the interrupt:</p> <p>0b00 Invalid (Inactive).</p> <p>0b01 Pending.</p> <p>0b10 Active.</p> <p>0b11 Pending and active.</p> <p>The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the invalid state are ignored, except for the purpose of generating virtual maintenance interrupts.</p> <p>For hardware interrupts, the pending and active state is held in the physical Distributor rather than the virtual CPU interface. A hypervisor must only use the pending and active state for software originated interrupts, which are typically associated with virtual devices, or SGIs.</p>	xx
[61]	HW	<p>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.</p> <p>0b0 The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated.</p> <p>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.</p> <p>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.</p>	x
[60]	Group	<p>Indicates the group for this virtual interrupt.</p> <p>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.</p> <p>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.</p> <p>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.</p>	x
[59:56]	RES0	Reserved	RES0
[55:48]	Priority	The priority of this interrupt. Bits [50:48] are RES0 .	8 {x}
[47:45]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[44:32]	pINTID	<p>Physical INTID, for hardware interrupts.</p> <p>When ICH_LR<n>_EL2.HW is 0 (there is no corresponding physical interrupt), this field has the following meaning:</p> <ul style="list-style-type: none"> 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. <p>When ICH_LR<n>_EL2.HW is 1 (there is a corresponding physical interrupt):</p> <ul style="list-style-type: none"> 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 valid 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. <p>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.</p>	13 {x}
[31:0]	vINTID	<p>Virtual INTID of the interrupt.</p> <p>If the value of vINTID is 1020-1023 and ICH_LR<n>_EL2.State!=0b00 (Inactive), behavior is UNPREDICTABLE.</p> <p>Behavior is UNPREDICTABLE if two or more List Registers specify the same vINTID when:</p> <ul style="list-style-type: none"> ICH_LR<n>_EL2.State == 0b01. ICH_LR<n>_EL2.State == 0b10. ICH_LR<n>_EL2.State == 0b11. <p>Bits [31:16] are RES0.</p> <p>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].</p>	32 {x}

Access

MRS <Xt>, ICH_LR<m>_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	'110':m[3]	m[2:0]

MSR ICH_LR<m>_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	'110':m[3]	m[2:0]

Accessibility

MRS <Xt>, ICH_LR<m>_EL2

```
integer m = UInt(CRm<0>:op2<2:0>);

if m >= 4 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICH_LR_EL2[m];
```

MSR ICH_LR<m>_EL2, <Xt>

```
integer m = UInt(CRm<0>:op2<2:0>);

if m >= 4 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    ICH_LR_EL2[m] = X[t, 64];
```

A.2.6.37 ICH_APOR<n>_EL2, Interrupt Controller Hyp Active Priorities Group 0 Registers, n = 0

Provides information about Group 0 virtual active priorities for EL2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC system registers

Access type

See bit descriptions

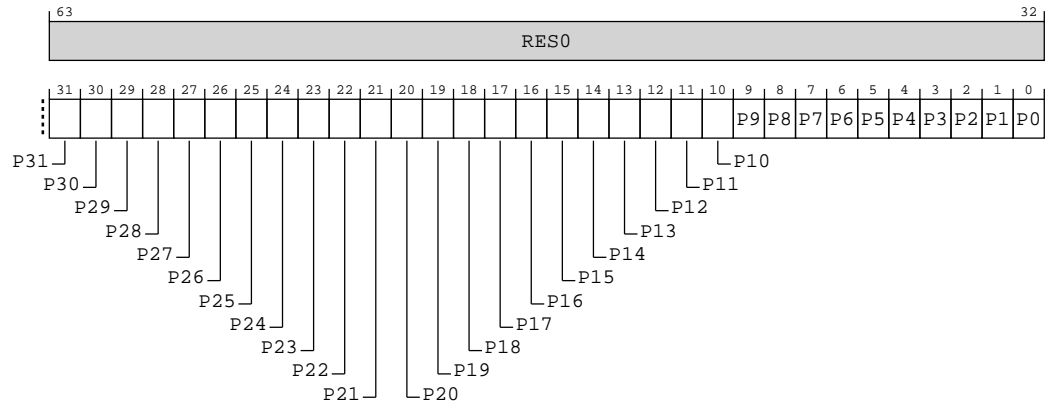
Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-355: AARCH64_ICH_AP0R<n>_EL2 bit assignments**Table A-899: ICH_AP0R<n>_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	P<x>	<p>Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are:</p> <p>0b0</p> <p>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.</p> <p>0b1</p> <p>There is a Group 0 interrupt active with this priority level which has not undergone priority drop.</p> <p>The correspondence between priority levels and bits depends on the number of bits of priority that are implemented.</p> <p>There are 32 preemption levels implemented, and the active state of these preemption levels are held in ICH_AP0R0_EL2 in the bits corresponding to Priority[7:3].</p> <p>Note:</p> <p>Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.</p>	0x00000000

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**.



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_AP0R<m>_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1000	'0':m[1:0]

MSR ICH_AP0R<m>_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1000	'0':m[1:0]

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**.



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_AP0R<m>_EL2

```
integer m = UInt(op2<1:0>);
if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICH_AP0R_EL2[m];
```

MSR ICH_AP0R<m>_EL2, <Xt>

```
integer m = UInt(op2<1:0>);
if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    ICH_AP0R_EL2[m] = X[t, 64];
```

A.2.6.38 ICH_AP1R<n>_EL2, Interrupt Controller Hyp Active Priorities Group 1 Registers, n = 0

Provides information about Group 1 virtual active priorities for EL2.

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	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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-356: AARCH64_ICH_AP1R<n>_EL2 bit assignments

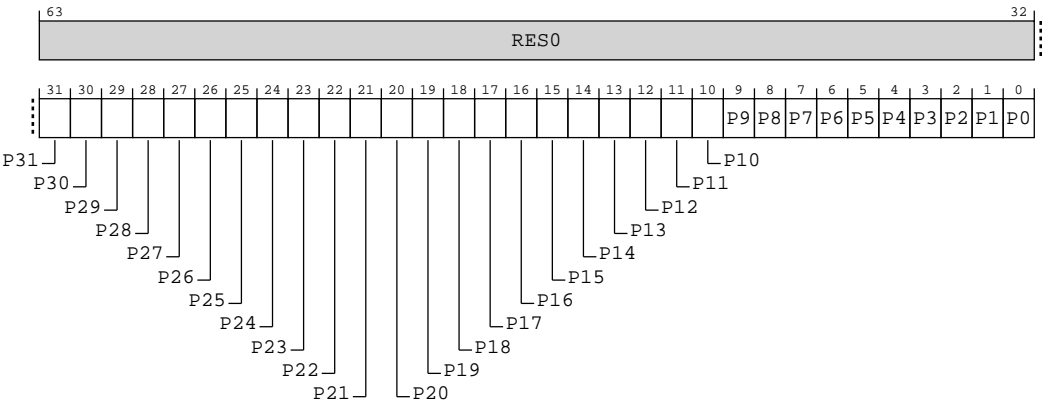


Table A-902: ICH_AP1R<n>_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	P<x>	<p>Group 1 interrupt active priorities. Possible values of each bit are:</p> <p>0b0</p> <p>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.</p> <p>0b1</p> <p>There is a Group 1 interrupt active with this priority level which has not undergone priority drop.</p> <p>The correspondence between priority levels and bits depends on the number of bits of priority that are implemented.</p> <p>There are 32 preemption levels implemented, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3].</p> <p>Note:</p> <p>Having the bit corresponding to a priority set to 1 in both ICH_AP0R<n>_EL2 and AArch64-ICH_AP1R<n>_EL2 might result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.</p>	0x00000000

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_AP1R<m>_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	'0':m[1:0]

MSR ICH_AP1R<m>_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	'0':m[1:0]

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.



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_AP1R<m>_EL2

```
integer m = UInt(op2<1:0>);
if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICH_AP1R_EL2[m];
```

MSR ICH_AP1R<m>_EL2, <Xt>

```
integer m = UInt(op2<1:0>);
if m == 1 then
    UNDEFINED;
elseif m == 2 || m == 3 then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    ICH_AP1R_EL2[m] = X[t, 64];
```

A.2.6.39 ICC_SRE_EL2, Interrupt Controller System Register Enable Register (EL2)

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

Configurations

This register is available in all configurations.

Attributes

Width

64

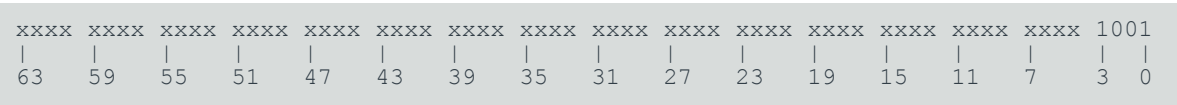
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-357: AARCH64_ICC_SRE_EL2 bit assignments

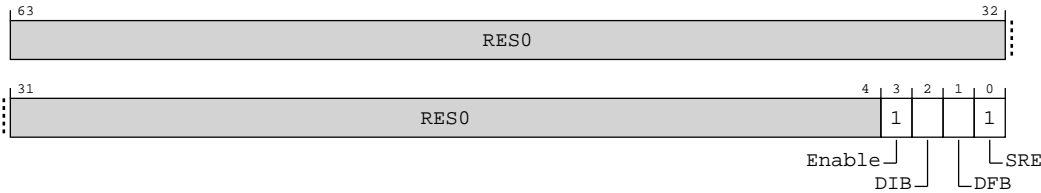


Table A-905: ICC_SRE_EL2 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3]	Enable	Enable. Enables lower Exception level access to AArch64-ICC_SRE_EL1. 0b1 EL1 accesses to AArch64-ICC_SRE_EL1 do not trap to EL2.	0b1

Bits	Name	Description	Reset
[2]	DIB	Disable IRQ bypass. 0b0 IRQ bypass enabled. 0b1 IRQ bypass disabled.	0b0
[1]	DFB	Disable FIQ bypass. 0b0 FIQ bypass enabled. 0b1 FIQ bypass disabled.	0b0
[0]	SRE	System Register Enable. 0b1 The System register interface to the ICH_* registers and the EL1 and EL2 ICC_* registers is enabled for EL2.	0b1

Access

Execution with AArch64-ICC_SRE_EL2.SRE set to 0 might make some System registers **UNKNOWN**.

MRS <Xt>, ICC_SRE_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b101

MSR ICC_SRE_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b101

Accessibility

Execution with AArch64-ICC_SRE_EL2.SRE set to 0 might make some System registers UNKNOWN.

MRS <Xt>, ICC_SRE_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ICC_SRE_EL2;
```

MSR ICC_SRE_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    ICC_SRE_EL2 = X[t, 64];
```

A.2.6.40 ICH_HCR_EL2, Interrupt Controller Hyp Control Register

Controls the environment for VMs.

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	0000	0xxx	xxxx	xxxx	x0x0	00xx	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-358: AARCH64_ICH_HCR_EL2 bit assignments

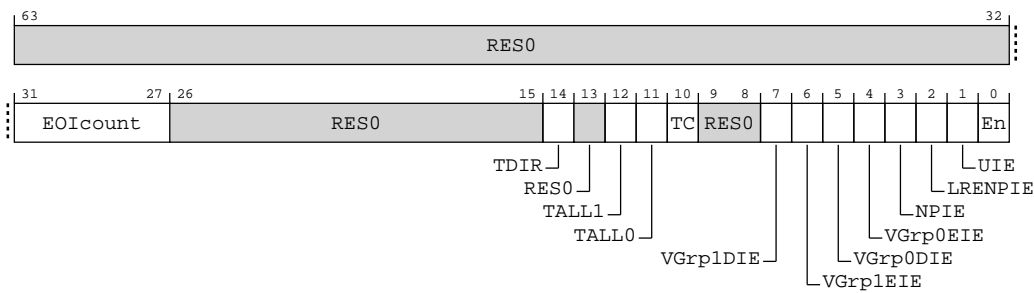


Table A-908: ICH_HCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:27]	EOIcount	<p>This field is incremented whenever a successful write to a virtual EOIR or DIR register would have resulted in a virtual interrupt deactivation. That is either:</p> <ul style="list-style-type: none"> A virtual write to EOIR with a valid interrupt identifier that is not in the LPI range (that is < 8192) when EOI mode is zero and no List Register was found. A virtual write to DIR with a valid interrupt identifier that is not in the LPI range (that is < 8192) when EOI mode is one and no List Register was found. <p>This allows software to manage more active interrupts than there are implemented List Registers.</p> <p>It is CONSTRAINED UNPREDICTABLE whether a virtual write to EOIR that does not clear a bit in the Active Priorities registers (AArch64-ICH_AP0R<n>_EL2/AArch64-ICH_AP1R<n>_EL2) increments EOIcount. The implemented behavior is to leave EOIcount unchanged.</p>	0b00000
[26:15]	RES0	Reserved	RES0
[14]	TDIR	<p>Trap EL1 writes to AArch64-ICC_DIR_EL1 and AArch64-ICV_DIR_EL1.</p> <p>0b0</p> <p>EL1 writes of AArch64-ICC_DIR_EL1 and AArch64-ICV_DIR_EL1 are not trapped to EL2, unless trapped by other mechanisms.</p> <p>0b1</p> <p>EL1 writes of AArch64-ICV_DIR_EL1 and AArch64-ICC_DIR_EL1 are trapped to EL2.</p>	0b0
[13]	RES0	Reserved	RES0
[12]	TALL1	<p>Trap all EL1 accesses to ICC_* and ICV_* System registers for Group 1 interrupts to EL2.</p> <p>0b0</p> <p>EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts proceed as normal.</p> <p>0b1</p> <p>EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts trap to EL2.</p>	0b0
[11]	TALLO	<p>Trap all EL1 accesses to ICC_* and ICV_* System registers for Group 0 interrupts to EL2.</p> <p>0b0</p> <p>EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts proceed as normal.</p> <p>0b1</p> <p>EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts trap to EL2.</p>	0b0
[10]	TC	<p>Trap all EL1 accesses to System registers that are common to Group 0 and Group 1 to EL2.</p> <p>0b0</p> <p>EL1 accesses to common registers proceed as normal.</p> <p>0b1</p> <p>EL1 accesses to common registers trap to EL2.</p> <p>This affects accesses to AArch64-ICC_SGI0R_EL1, AArch64-ICC_SGI1R_EL1, AArch64-ICC_ASGI1R_EL1, AArch64-ICC_CTLR_EL1, AArch64-ICC_DIR_EL1, AArch64-ICC_PMR_EL1, AArch64-ICC_RPR_EL1, AArch64-ICV_CTLR_EL1, AArch64-ICV_DIR_EL1, AArch64-ICV_PMR_EL1, and AArch64-ICV_RPR_EL1.</p>	0b0
[9:8]	RES0	Reserved	RES0
[7]	VGrp1DIE	<p>VM Group 1 Disabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected vPE is disabled:</p> <p>0b0</p> <p>Maintenance interrupt disabled.</p> <p>0b1</p> <p>Maintenance interrupt signaled when AArch64-ICH_VMCR_EL2.VENG1 is 0.</p>	0b0

Bits	Name	Description	Reset
[6]	VGrp1EIE	<p>VM Group 1 Enabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected vPE is enabled:</p> <p>0b0 Maintenance interrupt disabled.</p> <p>0b1 Maintenance interrupt signaled when AArch64-ICH_VMCR_EL2.VENG1 is 1.</p>	0b0
[5]	VGrp0DIE	<p>VM Group 0 Disabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected vPE is disabled:</p> <p>0b0 Maintenance interrupt disabled.</p> <p>0b1 Maintenance interrupt signaled when AArch64-ICH_VMCR_EL2.VENG0 is 0.</p>	0b0
[4]	VGrp0EIE	<p>VM Group 0 Enabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected vPE is enabled:</p> <p>0b0 Maintenance interrupt disabled.</p> <p>0b1 Maintenance interrupt signaled when AArch64-ICH_VMCR_EL2.VENG0 is 1.</p>	0b0
[3]	NPIE	<p>No Pending Interrupt Enable. Enables the signaling of a maintenance interrupt when there are no List registers with the State field set to 0b01 (pending):</p> <p>0b0 Maintenance interrupt disabled.</p> <p>0b1 Maintenance interrupt signaled while the List registers contain no interrupts in the pending state.</p>	0b0
[2]	LRENPIE	<p>List Register Entry Not Present Interrupt Enable. Enables the signaling of a maintenance interrupt while the virtual CPU interface does not have a corresponding valid List register entry for an EOI request:</p> <p>0b0 Maintenance interrupt disabled.</p> <p>0b1 Maintenance interrupt is asserted while the EOICount field is not 0.</p>	0b0
[1]	UIE	<p>Underflow Interrupt Enable. Enables the signaling of a maintenance interrupt when the List registers are empty, or hold only one valid entry:</p> <p>0b0 Maintenance interrupt disabled.</p> <p>0b1 Maintenance interrupt is asserted if none, or only one, of the List register entries is marked as a valid interrupt.</p>	0b0

Bits	Name	Description	Reset
[0]	En	<p>Enable. Global enable bit for the virtual CPU interface:</p> <p>0b0 Virtual CPU interface operation disabled.</p> <p>0b1 Virtual CPU interface operation enabled.</p> <p>When this field is set to 0:</p> <ul style="list-style-type: none"> The virtual CPU interface does not signal any maintenance interrupts. The virtual CPU interface does not signal any virtual interrupts. A read of AArch64-ICV_IAR0_EL1, AArch64-ICV_IAR1_EL1, ext-GICV_IAR or ext-GICV_AIAR returns a spurious interrupt ID. 	0b0

Access

MRS <Xt>, ICH_HCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b000

MSR ICH_HCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b000

Accessibility

MRS <Xt>, ICH_HCR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICH_HCR_EL2;

```

MSR ICH_HCR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    ICH_HCR_EL2 = X[t, 64];

```


A.2.6.41 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualization features.

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	1001	0000	0011	10xx	xxxx	xxxx	xxx0	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-359: AARCH64_ICH_VTR_EL2 bit assignments

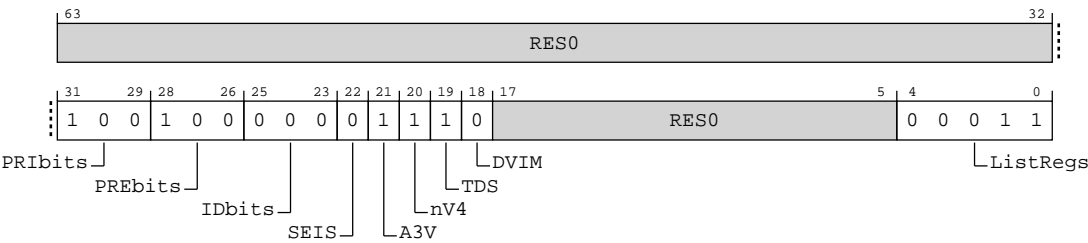


Table A-911: ICH_VTR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:29]	PRIbits	<p>Priority bits. The number of virtual priority bits implemented, minus one.</p> <p>An implementation must implement at least 32 levels of virtual priority (5 priority bits).</p> <p>This field is an alias of AArch64-ICV_CTLR_EL1.PRIbits.</p> <p>0b100</p> <p>5 virtual priority bits are implemented.</p>	0b100
[28:26]	PREbits	<p>The number of virtual preemption bits implemented, minus one. This field determines the minimum value of AArch64-ICH_VMCR_EL2.VBPR0.</p> <p>0b100</p> <p>5 virtual preemption bits are implemented.</p>	0b100
[25:23]	IDbits	<p>The number of virtual interrupt identifier bits supported:</p> <p>0b000</p> <p>16 bits.</p> <p>All other values are reserved.</p> <p>This field is an alias of AArch64-ICV_CTLR_EL1.IDbits.</p>	0b000
[22]	SEIS	<p>SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs:</p> <p>0b0</p> <p>The virtual CPU interface logic does not support generation of SEIs.</p> <p>This bit is an alias of AArch64-ICV_CTLR_EL1.SEIS.</p>	0b0
[21]	A3V	<p>Affinity 3 Valid. Possible values are:</p> <p>0b1</p> <p>The virtual CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.</p> <p>This bit is an alias of AArch64-ICV_CTLR_EL1.A3V.</p>	0b1
[20]	nV4	<p>Direct injection of virtual interrupts not supported. Possible values are:</p> <p>0b1</p> <p>The CPU interface logic does not support direct injection of virtual interrupts.</p>	0b1
[19]	TDS	<p>Separate trapping of EL1 writes to AArch64-ICV_DIR_EL1 supported.</p> <p>0b1</p> <p>Implementation supports AArch64-ICH_HCR_EL2.TDIR.</p>	0b1
[18]	DVIM	<p>Masking of directly-injected virtual interrupts.</p> <p>0b0</p> <p>Masking of Directly-injected Virtual Interrupts not supported.</p>	0b0
[17:5]	RES0	Reserved	RES0
[4:0]	ListRegs	<p>The number of implemented List registers, minus one.</p> <p>0b00011</p> <p>4 List registers implemented.</p>	0b00011

Access

MRS <Xt>, ICH_VTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b001

Accessibility

MRS <Xt>, ICH_VTR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ICH_VTR_EL2;
```

A.2.6.42 ICH_MISR_EL2, Interrupt Controller Maintenance Interrupt State Register

Indicates which maintenance interrupts are asserted.

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	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-360: AARCH64_ICH_MISR_EL2 bit assignments

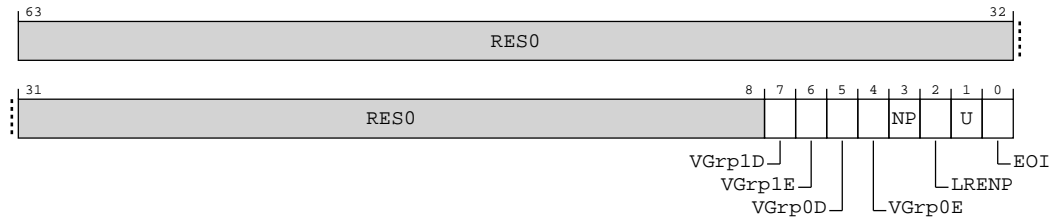


Table A-913: ICH_MISR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7]	VGrp1D	<p>vPE Group 1 Disabled.</p> <p>0b0 vPE Group 1 Disabled maintenance interrupt not asserted.</p> <p>0b1 vPE Group 1 Disabled maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when AArch64-ICH_HCR_EL2.VGrp1DIE==1 and AArch64-ICH_VMCR_EL2.VENG1==is 0.</p>	0b0
[6]	VGrp1E	<p>vPE Group 1 Enabled.</p> <p>0b0 vPE Group 1 Enabled maintenance interrupt not asserted.</p> <p>0b1 vPE Group 1 Enabled maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when AArch64-ICH_HCR_EL2.VGrp1EIE==1 and AArch64-ICH_VMCR_EL2.VENG1==is 1.</p>	0b0
[5]	VGrp0D	<p>vPE Group 0 Disabled.</p> <p>0b0 vPE Group 0 Disabled maintenance interrupt not asserted.</p> <p>0b1 vPE Group 0 Disabled maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when AArch64-ICH_HCR_EL2.VGrp0DIE==1 and AArch64-ICH_VMCR_EL2.VENG0==0.</p>	0b0
[4]	VGrp0E	<p>vPE Group 0 Enabled.</p> <p>0b0 vPE Group 0 Enabled maintenance interrupt not asserted.</p> <p>0b1 vPE Group 0 Enabled maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when AArch64-ICH_HCR_EL2.VGrp0EIE==1 and AArch64-ICH_VMCR_EL2.VENG0==1.</p>	0b0

Bits	Name	Description	Reset
[3]	NP	<p>No Pending.</p> <p>0b0 No Pending maintenance interrupt not asserted.</p> <p>0b1 No Pending maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when AArch64-ICH_HCR_EL2.NPIE==1 and no List register is in pending state.</p>	0b0
[2]	LRENP	<p>List Register Entry Not Present.</p> <p>0b0 List Register Entry Not Present maintenance interrupt not asserted.</p> <p>0b1 List Register Entry Not Present maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when AArch64-ICH_HCR_EL2.LRENPIE==1 and AArch64-ICH_HCR_EL2.EOIcount is nonzero.</p>	0b0
[1]	U	<p>Underflow.</p> <p>0b0 Underflow maintenance interrupt not asserted.</p> <p>0b1 Underflow maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when AArch64-ICH_HCR_EL2.UIE==1 and zero or one of the List register entries are marked as a valid interrupt, that is, if the corresponding AArch64-ICH_LR<n>_EL2.State bits do not equal 0x0.</p>	0b0
[0]	EOI	<p>End Of Interrupt.</p> <p>0b0 End Of Interrupt maintenance interrupt not asserted.</p> <p>0b1 End Of Interrupt maintenance interrupt asserted.</p> <p>This maintenance interrupt is asserted when at least one bit in AArch64-ICH_EISR_EL2 is 1.</p>	0b0

The U and NP bits do not include the status of any pending/active 'VSet (IRI)' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069) packets because these bits control generation of interrupts that allow software management of the contents of the List Registers (which are not affected by 'VSet (IRI)' packets).

Access

MRS <Xt>, ICH_MISR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b010

Accessibility

MRS <Xt>, ICH_MISR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ICH_MISR_EL2;
```

A.2.6.43 ICH_EISR_EL2, Interrupt Controller End of Interrupt Status Register

Indicates which List registers have outstanding EOI maintenance interrupts.

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-361: AARCH64_ICH_EISR_EL2 bit assignments

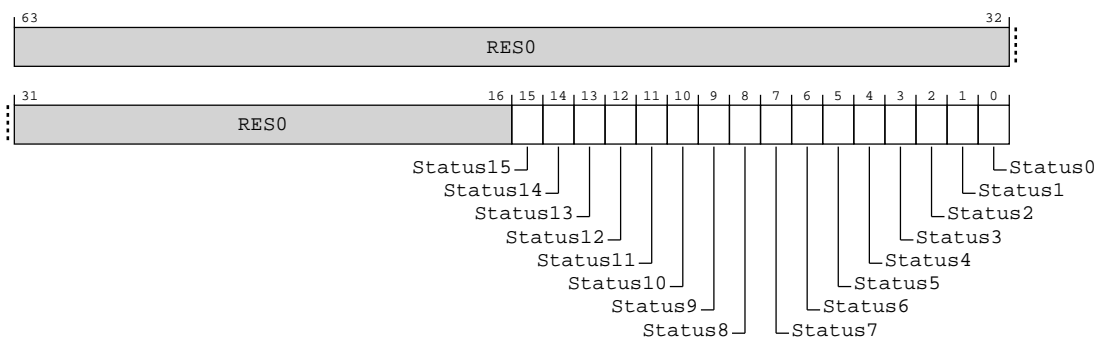


Table A-915: ICH_EISR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:0]	Status<n>	<div>EOI maintenance interrupt status bit for List register <n>: 0b0 List register <n>, AArch64-ICH_LR<n>_EL2, does not have an EOI maintenance interrupt. 0b1 List register <n>, AArch64-ICH_LR<n>_EL2, has an EOI maintenance interrupt that has not been handled. For any AArch64-ICH_LR<n>_EL2, the corresponding status bit is set to 1 if all of the following are true:<ul style="list-style-type: none">AArch64-ICH_LR<n>_EL2.State is 0b00.AArch64-ICH_LR<n>_EL2.HW is 0.AArch64-ICH_LR<n>_EL2.EOI (bit [41]) is 1, indicating that when the interrupt corresponding to that List register is deactivated, a maintenance interrupt is asserted.Otherwise the status bit takes the value 0.</div>	16{x}

Access

MRS <Xt>, ICH_EISR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b011

Accessibility

MRS <Xt>, ICH_EISR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICH_EISR_EL2;
```

A.2.6.44 ICH_ELRSR_EL2, Interrupt Controller Empty List Register Status Register

These registers can be used to locate a usable List register when the hypervisor is delivering an interrupt to a VM.

Configurations

This register is available in all configurations.

Attributes

Width

64

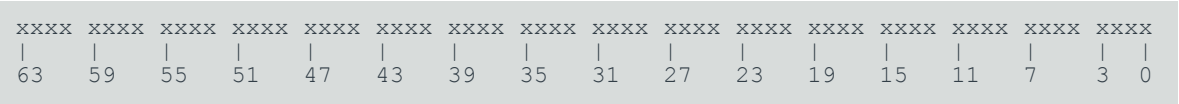
Functional group

GIC system registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-362: AARCH64_ICH_ELRSR_EL2 bit assignments

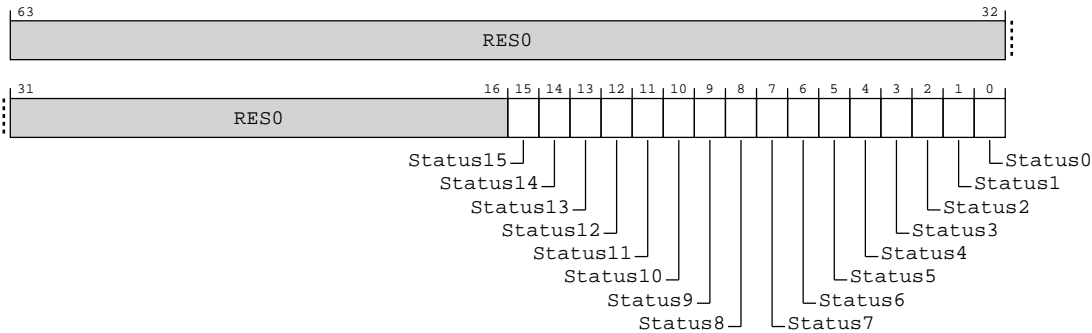


Table A-917: ICH_ELRSR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15:0]	Status<n>	<p>Status bit for List register <n>, AArch64-ICH_LR<n>_EL2:</p> <p>0b0</p> <p>List register AArch64-ICH_LR<n>_EL2, if implemented, contains a valid interrupt. Using this List register can result in overwriting a valid interrupt.</p> <p>0b1</p> <p>List register AArch64-ICH_LR<n>_EL2 does not contain a valid interrupt. The List register is empty and can be used without overwriting a valid interrupt or losing an EOI maintenance interrupt.</p> <p>For any List register <n>, the corresponding status bit is set to 1 if AArch64-ICH_LR<n>_EL2.State is 0b00 and either AArch64-ICH_LR<n>_EL2.HW is 1 or AArch64-ICH_LR<n>_EL2.EOI (bit [41]) is 0.</p> <p>Otherwise the status bit takes the value 0.</p>	16 {x}

Access

MRS <Xt>, ICH_ELRSR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b101

Accessibility

MRS <Xt>, ICH_ELRSR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICH_ELRSR_EL2;

```

A.2.6.45 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register

Enables the hypervisor to save and restore the virtual machine view of the GIC state.

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	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-363: AARCH64_ICH_VMCR_EL2 bit assignments

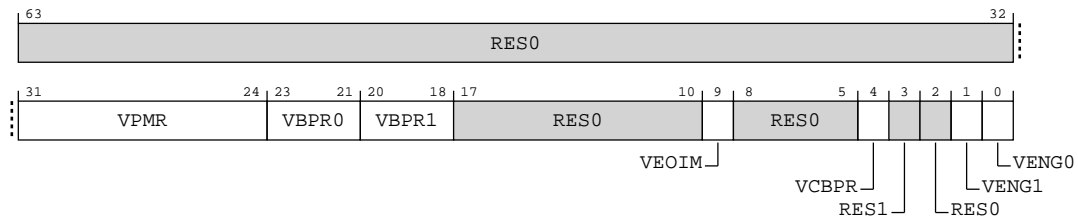


Table A-919: ICH_VMCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	VPMR	Virtual Priority Mask. The priority mask level for the virtual CPU interface. If the priority of a pending virtual interrupt is higher than the value indicated by this field, the interface signals the virtual interrupt to the PE. This field is an alias of AArch64-ICV_PMR_EL1.Priority.	8 {x}
[23:21]	VBPR0	Virtual Binary Point Register, Group 0. Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption, and also determines Group 1 interrupt preemption if ICH_VMCR_EL2.VCBPR == 1. This field is an alias of AArch64-ICV_BPR0_EL1.BinaryPoint. The minimum value of this field is determined by AArch64-ICH_VTR_EL2.PREbits. An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.	xxx
[20:18]	VBPR1	Virtual Binary Point Register, Group 1. Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption if AArch64-ICH_VMCR_EL2.VCBPR == 0. This field is an alias of AArch64-ICV_BPR1_EL1.BinaryPoint. This field is always accessible to EL2 accesses, regardless of the setting of the ICH_VMCR_EL2.VCBPR field. For Secure writes, the minimum value of this field is the minimum value of ICH_VMCR_EL2.VBPR0. An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.	xxx

Bits	Name	Description	Reset
[17:10]	RES0	Reserved	RES0
[9]	VEOIM	<p>Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:</p> <p>0b0</p> <p>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.</p> <p>0b1</p> <p>AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.</p> <p>This bit is an alias of AArch64-ICV_CTLR_EL1.EOI mode.</p>	x
[8:5]	RES0	Reserved	RES0
[4]	VCBPR	<p>Virtual Common Binary Point Register. Possible values of this bit are:</p> <p>0b0</p> <p>AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.</p> <p>0b1</p> <p>Reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Writes to AArch64-ICV_BPR1_EL1 are ignored.</p> <p>This field is an alias of AArch64-ICV_CTLR_EL1.CBPR.</p>	x
[3]	RES1	Reserved	RES1
[2]	RES0	Reserved	RES0
[1]	VENG1	<p>Virtual Group 1 interrupt enable. Possible values of this bit are:</p> <p>0b0</p> <p>Virtual Group 1 interrupts are disabled.</p> <p>0b1</p> <p>Virtual Group 1 interrupts are enabled.</p> <p>This bit is an alias of AArch64-ICV_IGRPEN1_EL1.Enable.</p>	x
[0]	VENG0	<p>Virtual Group 0 interrupt enable. Possible values of this bit are:</p> <p>0b0</p> <p>Virtual Group 0 interrupts are disabled.</p> <p>0b1</p> <p>Virtual Group 0 interrupts are enabled.</p> <p>This bit is an alias of AArch64-ICV_IGRPEN0_EL1.Enable.</p>	x

Access

When EL2 is using System register access, EL1 using either System register or memory-mapped access must be supported.

MRS <Xt>, ICH_VMCR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b111

MSR ICH_VMCR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1011	0b111

Accessibility

When EL2 is using System register access, EL1 using either System register or memory-mapped access must be supported.

MRS <Xt>, ICH_VMCR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ICH_VMCR_EL2;
```

MSR ICH_VMCR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    ICH_VMCR_EL2 = X[t, 64];
```

A.2.7 AArch64 RAS register description

This section includes the register descriptions for all *Reliability, Availability, and Serviceability* (RAS) registers in the Cortex®-R82 processor.

A.2.7.1 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

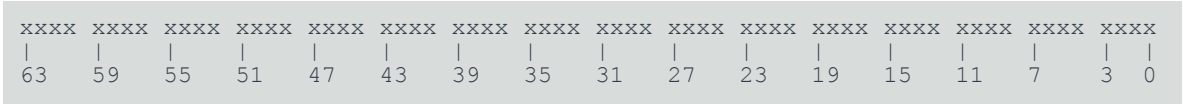
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-364: AARCH64_ERRIDR_EL1 bit assignments

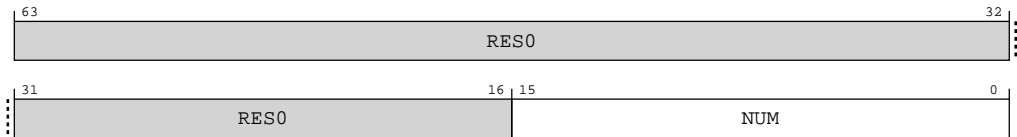


Table A-922: ERRIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:0]	NUM	<p>When RAM_PROTECTION == 1</p> <p>Highest numbered index of the records that can be accessed through the Error Record System registers plus one.</p> <p>0000000000000111</p> <p>7 error records implemented.</p> <p>When RAM_PROTECTION == 0</p> <p>Highest numbered index of the records that can be accessed through the Error Record System registers plus one.</p> <p>0000000000000001</p> <p>1 error record implemented.</p> <p>Otherwise</p> <p>UNKNOWN</p>	16{x}

Access

MRS <Xt>, ERRIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b000

Accessibility

MRS <Xt>, ERRIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERRIDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ERRIDR_EL1;
```

A.2.7.2 ERRSELR_EL1, Error Record Select Register

Selects an error record to be accessed through the Error Record System registers.

Configurations

If AArch64-ERRIDR_EL1 indicates that zero error records are implemented, then it is IMPLEMENTATION DEFINED whether ERRSELR_EL1 is UNDEFINED or RES0.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-365: AARCH64_ERRSELR_EL1 bit assignments

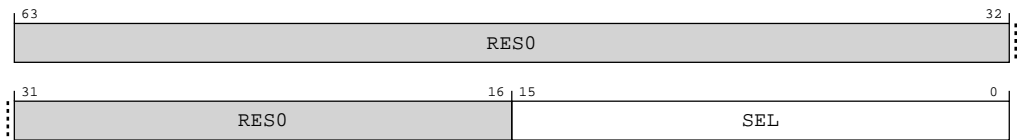


Table A-924: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15:0]	SEL	<p>Selects the error record accessed through the ERX registers.</p> <p>0b0000000000000000 Select record 0, containing non-ECC memory errors from L2 cache and LCU.</p> <p>0b0000000000000001 Select record 1, containing memory errors from L1 I-cache, L1 D-cache, ITCM, DTCM and MMS.</p> <p>0b0000000000000010 Select record 2, containing memory errors from L1 I-cache, L1 D-cache, ITCM, DTCM and MMS.</p> <p>0b0000000000000011 Select record 3, containing memory errors from L1 I-cache, L1 D-cache, ITCM, DTCM and MMS.</p> <p>0b0000000000000100 Select record 4, containing memory errors from L2 cache and LCU.</p> <p>0b0000000000000101 Select record 5, containing memory errors from L2 cache and LCU.</p> <p>0b0000000000000110 Select record 6, containing memory errors from L2 cache and LCU.</p> <p>For example, if ERRSELR_EL1.SEL is 0x0004, then direct reads and writes of AArch64-ERXSTATUS_EL1 access ERR4STATUS.</p> <p>If ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then all of the following apply:</p> <ul style="list-style-type: none">• The value read back from ERRSELR_EL1.SEL is UNKNOWN.• One of the following occurs:<ul style="list-style-type: none">◦ An UNKNOWN error record is selected.◦ The ERX*_EL1 registers are RAZ/WI.	16{x}

Access

MRS <Xt>, ERRSELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b001

MSR ERRSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0011	0b001

Accessibility

MRS <Xt>, ERRSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERRSELR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ERRSELR_EL1;
```

MSR ERRSELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERRSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ERRSELR_EL1 = X[t, 64];
```

A.2.7.3 ERXFR_EL1, Selected Error Record Feature Register

Accesses ext-ERR<n>FR for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-366: AARCH64_ERXFR_EL1 bit assignments

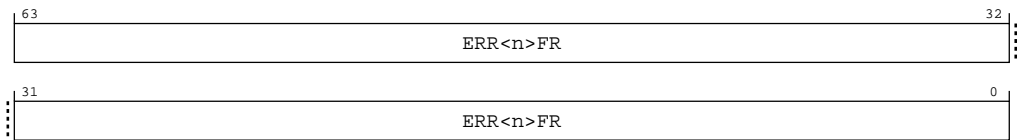


Table A-927: ERXFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXFR_EL1 accesses ext-ERR<n>FR, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 {x}

Access

If 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.

MRS <Xt>, ERXFR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b000

Accessibility

If 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.

MRS <Xt>, ERXFR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXFR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ERXFR_EL1;
```

A.2.7.4 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

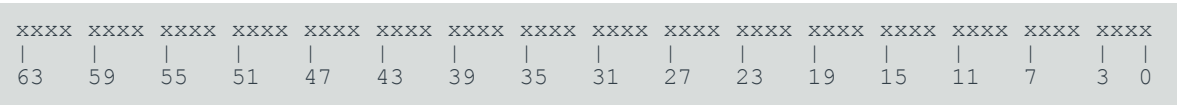
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-367: AARCH64_ERXCTLR_EL1 bit assignments

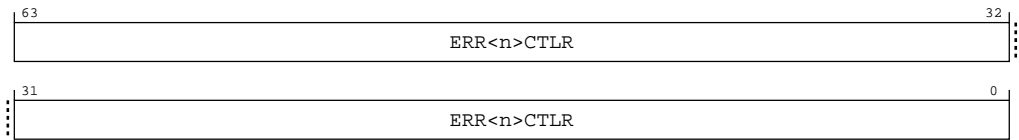


Table A-929: ERXCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXCTLR_EL1 accesses ext-ERR<n>CTLR, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 { x }

Access

If 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.

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-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.

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 HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXCTLR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ERXCTLR_EL1;

```

MSR ERXCTLR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXCTLR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ERXCTLR_EL1 = X[t, 64];

```

A.2.7.5 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

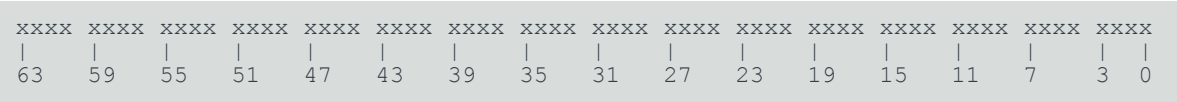
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-368: AARCH64_ERXSTATUS_EL1 bit assignments

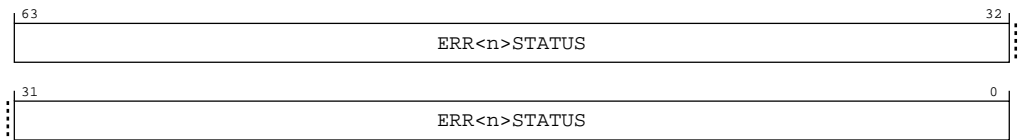


Table A-932: ERXSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXSTATUS_EL1 accesses ext-ERR<n>STATUS, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 { x }

Access

If 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.

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-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.

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;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXSTATUS_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ERXSTATUS_EL1;
```

MSR ERXSTATUS_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXSTATUS_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ERXSTATUS_EL1 = X[t, 64];
```

A.2.7.6 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

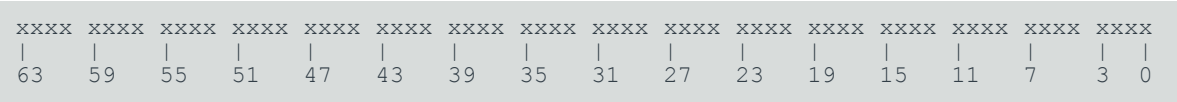
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-369: AARCH64_ERXADDR_EL1 bit assignments

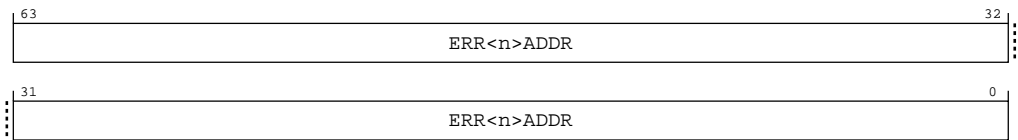


Table A-935: ERXADDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXADDR_EL1 accesses ext-ERR<n>ADDR, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 { x }

Access

If AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXADDR_EL1 is **RAZ/WI**.

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-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXADDR_EL1 is RAZ/WI.

MRS <Xt>, ERXADDR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXADDR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ERXADDR_EL1;

```

MSR ERXADDR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXADDR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ERXADDR_EL1 = X[t, 64];

```

A.2.7.7 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

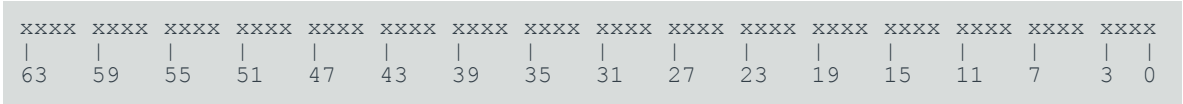
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-370: AARCH64_ERXPFGF_EL1 bit assignments

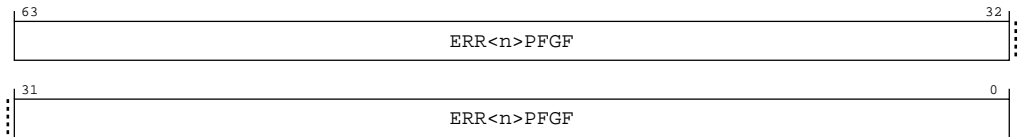


Table A-938: ERXPFGF_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXPFGF_EL1 accesses ext-ERR<n>PFGF, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 {x}

Access

If 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.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then direct reads of ERXPFGF_EL1 are NOPs.



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-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.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then direct reads of ERXPFGF_EL1 are NOPs.



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;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXPFGF_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ERXPFGF_EL1;

```

A.2.7.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control Register

Accesses ext-ERR<n>PFGCTL for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-371: AARCH64_ERXPGCTL_EL1 bit assignments

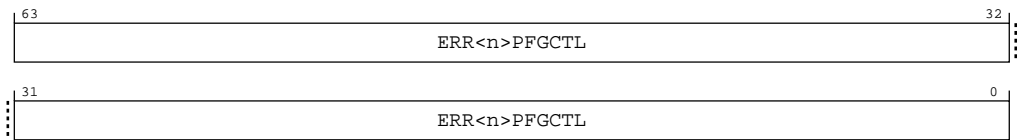


Table A-940: ERXPGCTL_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXPGCTL_EL1 accesses ext-ERR<n>PFGCTL, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 {x}

Access

If 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.
- ERXPGCTL_EL1 is RAZ/WI.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then direct reads and writes of ERXPGCTL_EL1 are NOPs.



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-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.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then direct reads and writes of ERXPFGCTL_EL1 are NOPs.



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 HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXPFGCTL_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = ERXPFGCTL_EL1;
```

MSR ERXPGCTL_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXPGCTL_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    ERXPGCTL_EL1 = X[t, 64];
```

A.2.7.9 ERXPGCDN_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-372: AARCH64_ERXPGCDN_EL1 bit assignments

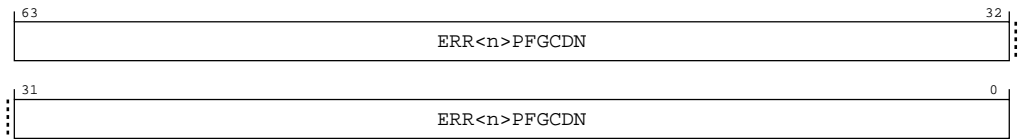


Table A-943: ERXPFGCDN_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXPFGCDN_EL1 accesses ext-ERR<n>PFGCDN, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 {x}

Access

If 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.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then direct reads and writes of ERXPFGCDN_EL1 are NOPs.

**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 **RESO**.

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-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.

If AArch64-ERRSELR_EL1.SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then direct reads and writes of ERXPFGCDN_EL1 are NOPs.



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;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXPFGCDN_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ERXPFGCDN_EL1;
```

MSR ERXPFGCDN_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXPFGCDN_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ERXPFGCDN_EL1 = X[t, 64];
```

A.2.7.10 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0

Accesses ext-ERR<n>MISCO for the error record <n> selected by AArch64-ERRSELR_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

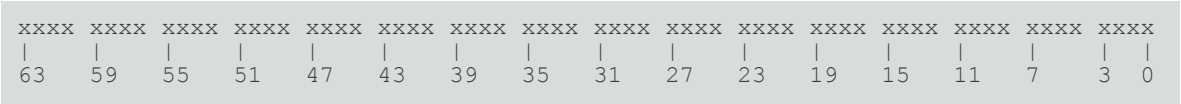
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-373: AARCH64_ERXMISC0_EL1 bit assignments

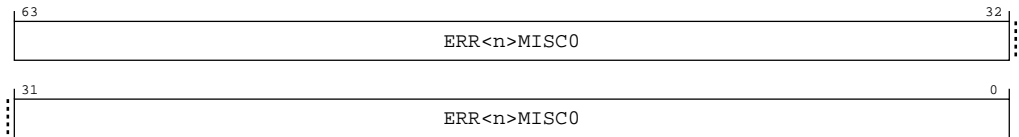


Table A-946: ERXMISC0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC0_EL1 accesses ext-ERR<n>MISC0, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 { x }

Access

If 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.

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-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.
- ERXMISCO_EL1 is RAZ/WI.

ext-ERR<n>MISCO describes additional constraints that also apply when ext-ERR<n>MISCO is accessed through ERXMISCO_EL1.

MRS <Xt>, ERXMISCO_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXMISCO_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ERXMISCO_EL1;

```

MSR ERXMISCO_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXMISCO_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ERXMISCO_EL1 = X[t, 64];

```

A.2.7.11 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

```




Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-374: AARCH64_ERXMISC1_EL1 bit assignments

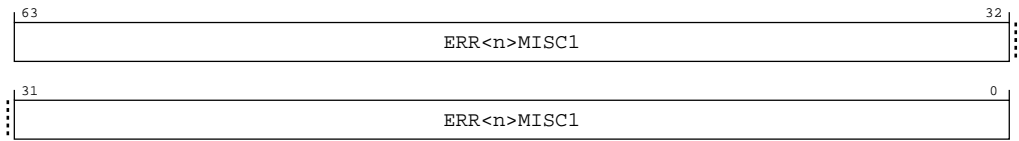


Table A-949: ERXMISC1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC1_EL1 accesses ext-ERR<n>MISC1, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 { x }

Access

If AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXMISC1_EL1 is **RAZ/WI**.

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-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXMISC1_EL1 is **RAZ/WI**.

MRS <Xt>, ERXMISC1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXMISC1_EL1;
    elsif PSTATE.EL == EL2 then
```

```
X[t, 64] = ERXMISC1_EL1;
```

MSR ERXMISC1_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXMISC1_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ERXMISC1_EL1 = X[t, 64];
```

A.2.7.12 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-375: AARCH64_ERXMISC2_EL1 bit assignments

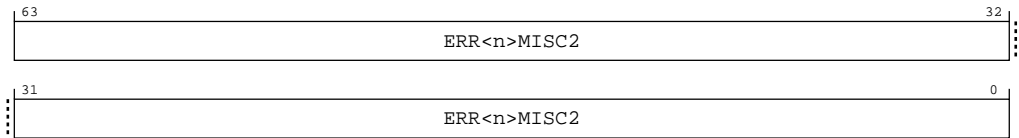


Table A-952: ERXMISC2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC2_EL1 accesses ext-ERR<n>MISC2, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 {x}

Access

If AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXMISC2_EL1 is **RAZ/WI**.

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-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXMISC2_EL1 is **RAZ/WI**.

MRS <Xt>, ERXMISC2_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXMISC2_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ERXMISC2_EL1;
```

MSR ERXMISC2_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```
    ERXMISC2_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ERXMISC2_EL1 = X[t, 64];
```

A.2.7.13 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

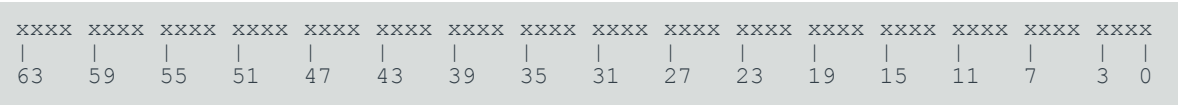
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-376: AARCH64_ERXMISC3_EL1 bit assignments

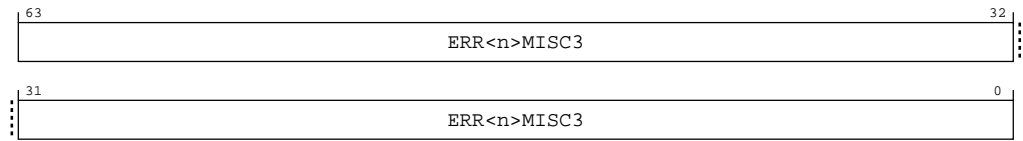


Table A-955: ERXMISC3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	ERXMISC3_EL1 accesses ext-ERR<n>MISC3, where <n> is the value in AArch64-ERRSELR_EL1.SEL.	64 { x }

Access

If AArch64-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXMISC3_EL1 is **RAZ/WI**.

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-ERRSELR_EL1.SEL is greater than or equal to AArch64-ERRIDR_EL1.NUM, then ERXMISC3_EL1 is RAZ/WI.

MRS <Xt>, ERXMISC3_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ERXMISC3_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ERXMISC3_EL1;
```

MSR ERXMISC3_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ERXMISC3_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ERXMISC3_EL1 = X[t, 64];
```

A.2.7.14 DISR_EL1, Deferred Interrupt Status Register

Records that an SError interrupt has been consumed by an `ESB` instruction.

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	xxx0	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-377: AARCH64_DISR_EL1 bit assignments

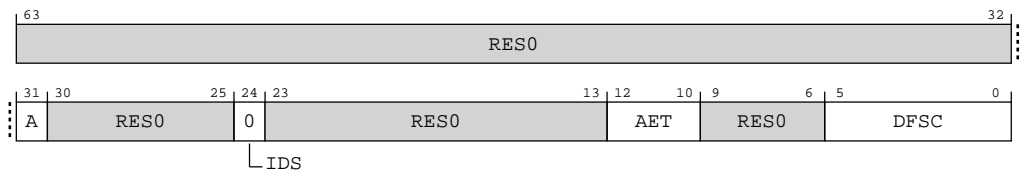


Table A-958: DISR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	A	Set to 1 when an ESB instruction defers an asynchronous SError interrupt.	x
[30:25]	RES0	Reserved	RES0
[24]	IDS	Indicates the deferred SError interrupt type. 0b0 Deferred error uses architecturally-defined format.	0b0
[23:13]	RES0	Reserved	RES0
[12:10]	AET	Asynchronous Error Type. See the description of ESR_ELx.AET for an SError interrupt.	xxx
[9:6]	RES0	Reserved	RES0
[5:0]	DFSC	Fault Status Code. See the description of ESR_ELx.DFSC for an SError interrupt.	6{x}

Access

An indirect write to DISR_EL1 made by an `ESB` instruction does not require an explicit synchronization operation for the value that is written to be observed by a direct read of DISR_EL1 occurring in program order after the `ESB` instruction.

MRS <Xt>, DISR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0001	0b001

MSR DISR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0001	0b001

Accessibility

An indirect write to DISR_EL1 made by an `ESB` instruction does not require an explicit synchronization operation for the value that is written to be observed by a direct read of DISR_EL1 occurring in program order after the `ESB` instruction.

MRS <Xt>, DISR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.AMO == '1' then
        X[ $\bar{t}$ , 64] = VDISR_EL2;
    else
        X[t, 64] = DISR_EL1;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = DISR_EL1;

```

MSR DISR_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.AMO == '1' then
        VDISR_EL2 = X[t, 64];
    else
        DISR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        DISR_EL1 = X[t, 64];

```

A.2.7.15 VESR_EL2, Virtual SError Exception Syndrome Register

Provides the syndrome value reported to software on taking a virtual SError interrupt exception to EL1, or on executing an `ESB` instruction at EL1.

When the virtual SError interrupt injected using AArch64-HCR_EL2.VSE is taken to EL1 using AArch64, then the syndrome value is reported in AArch64-ESR_EL1.

When the virtual SError interrupt injected using AArch64-HCR_EL2.VSE is deferred by an `ESB` instruction, then the syndrome value is written to AArch64-VDISR_EL2.

Configurations

This register is available in all configurations.

Attributes

Width

64

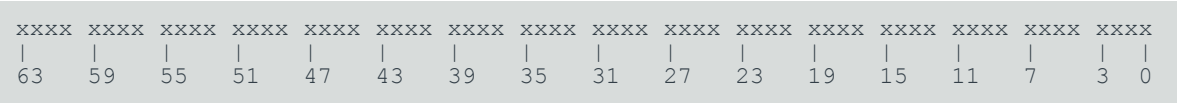
Functional group

RAS registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-378: AARCH64_VSESR_EL2 bit assignments

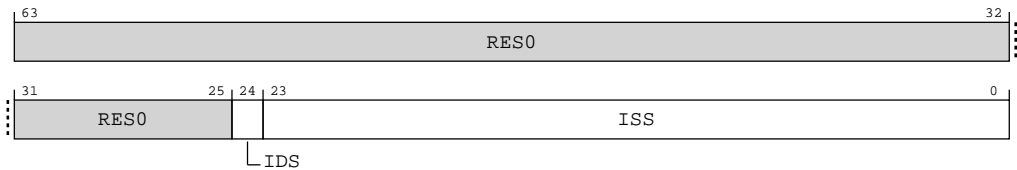


Table A-961: VSESR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:25]	RES0	Reserved	RES0
[24]	IDS	When a virtual SError interrupt is taken to EL1 using AArch64, AArch64-ESR_EL1[24] is set to VSESR_EL2.IDS. When a virtual SError interrupt is deferred by an <code>ESB</code> instruction, AArch64-VDISR_EL2[24] is set to VSESR_EL2.IDS.	x
[23:0]	ISS	When a virtual SError interrupt is taken to EL1 using AArch64, AArch64-ESR_EL1[23:0] is set to VSESR_EL2.ISS. When a virtual SError interrupt is deferred by an <code>ESB</code> instruction, AArch64-VDISR_EL2[23:0] is set to VSESR_EL2.ISS.	24 { x }

Access

MRS <Xt>, VSESR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b011

MSR VSESR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b011

Accessibility

MRS <Xt>, VSESR_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = VSESR_EL2;

```

MSR VSESR_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    VSESR_EL2 = X[t, 64];

```

A.2.7.16 VDISR_EL2, Virtual Deferred Interrupt Status Register

Records that a virtual SError interrupt has been consumed by an **ESB** instruction executed at EL1.

An indirect write to VDISR_EL2 made by an **ESB** instruction does not require an explicit synchronization operation for the value written to be observed by a direct read of one of the following registers occurring in program order after the **ESB** instruction:

- AArch64-DISR_EL1.

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-379: AArch64_VDISR_EL2 bit assignments

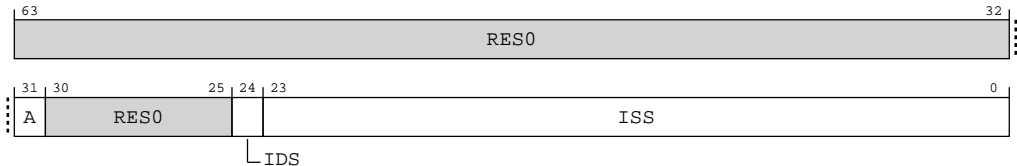


Table A-964: VDISR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	A	Set to 1 when an ESB instruction defers a virtual SError interrupt.	x
[30:25]	RES0	Reserved	RES0
[24]	IDS	The value copied from AArch64-VSESR_EL2.IDS.	x
[23:0]	ISS	The value copied from AArch64-VSESR_EL2.ISS.	24 { x }

Access

An indirect write to VDISR_EL2 made by an ESB instruction does not require an explicit synchronization operation for the value that is written to be observed by a direct read of one of the following registers occurring in program order after the ESB instruction:

- AArch64-DISR_EL1.

MRS <Xt>, VDISR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0001	0b001

MSR VDISR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0001	0b001

MRS <Xt>, DISR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0001	0b001

MSR DISR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0001	0b001

Accessibility

An indirect write to VDISR_EL2 made by an `ESB` instruction does not require an explicit synchronization operation for the value that is written to be observed by a direct read of one of the following registers occurring in program order after the `ESB` instruction:

- AArch64-DISR_EL1.

MRS <Xt>, VDISR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VDISR_EL2;
```

MSR VDISR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    VDISR_EL2 = X[t, 64];
```

MRS <Xt>, DISR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HCR_EL2.AMO == '1' then
        X[t, 64] = VDISR_EL2;
    else
        X[t, 64] = DISR_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = DISR_EL1;
```

MSR DISR_EL1, <Xt>

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

```
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if HCR_EL2.AMO == '1' then
        VDISR_EL2 = X[t, 64];
    else
        DISR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    DISR_EL1 = X[t, 64];
```

A.2.8 AArch64 Trace register description

This section includes the register descriptions for all Trace registers in the Cortex®-R82 processor.

A.2.8.1 TRCSEQEVR<n>, Sequencer State Transition Control Register <n> , n = 0 - 2

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

AArch64 register TRCSEQEVR<n> bits [31:0] are architecturally mapped to External register [B.2.2.7.19 TRCSEQEVR<n>, Sequencer State Transition Control Register <n> , n = 0 - 2](#) on page 2011 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-380: AARCH64_TRCSEQEVR<n> bit assignments

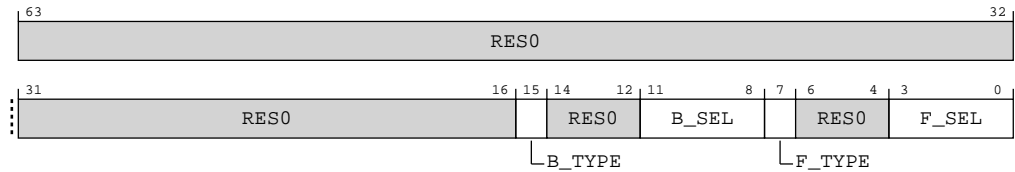


Table A-969: TRCSEQEVR<n> bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	RES0
[15]	B_TYPE	<p>Chooses the type of Resource Selector.</p> <p>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.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>B_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>B_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. B_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[14:12]	RES0	Reserved	RES0
[11:8]	B_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. B_TYPE controls whether B_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>Backward field. Selects the single Resource Selector or Resource Selector pair.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx

Bits	Name	Description	Reset
[7]	F_TYPE	Chooses the type of Resource Selector. Forward 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. F_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. F_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. F_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0
[3:0]	F_SEL	Defines the selected Resource Selector or pair of Resource Selectors. F_TYPE controls whether F_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. Forward field. Selects the single Resource Selector or Resource Selector pair. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire.	xxxx

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

MRS <Xt>, TRCSEQEVR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'00':m[1:0]	0b100

MSR TRCSEQEVR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'00':m[1:0]	0b100

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

MRS <Xt>, TRCSEQEVR<m>

```
integer m = UInt(CRm<1:0>);
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
```

```

    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSEQEVR[m];
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = TRCSEQEVR[m];

```

MSR TRCSEQEVR<m>, <Xt>

```

integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSEQEVR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSEQEVR[m] = X[t, 64];

```

A.2.8.2 TRCCNTRLDVR<n>, Counter Reload Value Register <n> , n = 0 - 1

This sets or returns the reload count value for counter <n>.

Configurations

AArch64 register TRCCNTRLDVR<n> bits [31:0] are architecturally mapped to External register [B.2.2.7.23 TRCCNTRLDVR<n>, Counter Reload Value Register <n> , n = 0 - 1](#) on page 2018 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-381: AARCH64_TRCCNTRLDVR<n> bit assignments

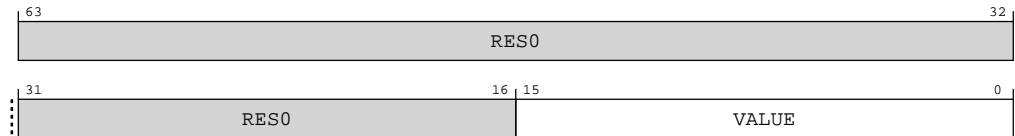


Table A-972: TRCCNTRLDVR<n> bit descriptions

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] == 0b1.

MRS <Xt>, TRCCNTRLDVR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'00':m[1:0]	0b101

MSR TRCCNTRLDVR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'00':m[1:0]	0b101

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 0b1.

MRS <Xt>, TRCCNTRLDVR<m>

```
integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```



```

    else
        X[t, 64] = TRCCNTRLDVR[m];
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = TRCCNTRLDVR[m];

```

MSR TRCCNTRLDVR<m>, <Xt>

```

integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCNTRLDVR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCNTRLDVR[m] = X[t, 64];

```

A.2.8.3 TRCCNTCTLR<n>, Counter Control Register <n> , n = 0 - 1

Controls the operation of counter <n>.

Configurations

AArch64 register TRCCNTCTLR<n> bits [31:0] are architecturally mapped to External register [B.2.2.7.24 TRCCNTCTLR<n>, Counter Control Register <n> , n = 0 - 1](#) on page 2020 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

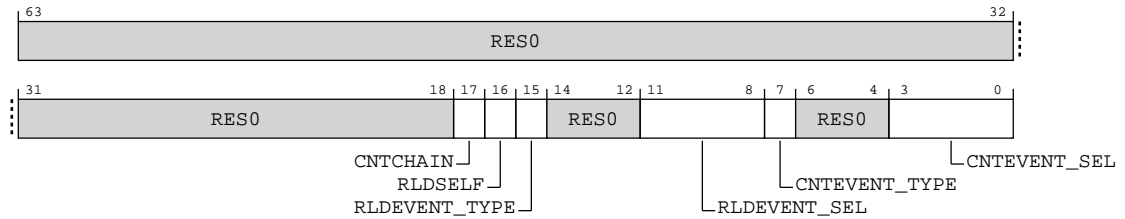
See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-382: AARCH64_TRCCNTCTLR<n> bit assignments****Table A-975: TRCCNTCTLR<n> bit descriptions**

Bits	Name	Description	Reset
[63:18]	RES0	Reserved	RES0
[17]	CNTCHAIN	For TRCCNTCTLR1, this bit controls whether the counter decrements when a reload event occurs for counter <n-1>. <p>0b0</p> The counter does not decrement when a reload event for counter <n-1> occurs. <p>0b1</p> Counter <n> decrements when a reload event for counter <n-1> occurs. This concatenates counter <n> and counter <n-1>, to provide a larger count value. CNTCHAIN is not implemented for TRCCNTCTLR0.	x
[16]	RLDSELF	Controls whether a reload event occurs for the counter, when the counter reaches zero. <p>0b0</p> Normal mode. The counter is in Normal mode. <p>0b1</p> Self-reload mode. The counter is in Self-reload mode.	x

Bits	Name	Description	Reset
[15]	RLDEVENT_TYPE	<p>Chooses the type of Resource Selector.</p> <p>Selects an event, that when it occurs causes a reload event for counter <n>.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>RLDEVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>RLDEVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. RLDEVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[14:12]	RES0	Reserved	RES0
[11:8]	RLDEVENT_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. RLDEVENT_TYPE controls whether RLDEVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>Selects an event, that when it occurs causes a reload event for counter <n>.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx
[7]	CNTEVENT_TYPE	<p>Chooses the type of Resource Selector.</p> <p>Selects an event, that when it occurs causes counter <n> to decrement.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>CNTEVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>CNTEVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. CNTEVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[6:4]	RES0	Reserved	RES0
[3:0]	CNTEVENT_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. CNTEVENT_TYPE controls whether CNTEVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>Selects an event, that when it occurs causes counter <n> to decrement.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 0b1.

MRS <Xt>, TRCCNTCTLR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'01':m[1:0]	0b101

MSR TRCCNTCTLR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'01':m[1:0]	0b101

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 0b1.

MRS <Xt>, TRCCNTCTLR<m>

```
integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[m];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCNTCTLR[m];
```

MSR TRCCNTCTLR<m>, <Xt>

```
integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCNTCTLR[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCNTCTLR[m] = X[t, 64];
```

A.2.8.4 TRCCNTVR<n>, Counter Value Register <n> , n = 0 - 1

This sets or returns the value of counter <n>.

Configurations

AArch64 register TRCCNTVR<n> bits [31:0] are architecturally mapped to External register B.2.2.7.25 TRCCNTVR<n>, Counter Value Register <n> , n = 0 - 1 on page 2023 bits [31:0].

Attributes

Width

64

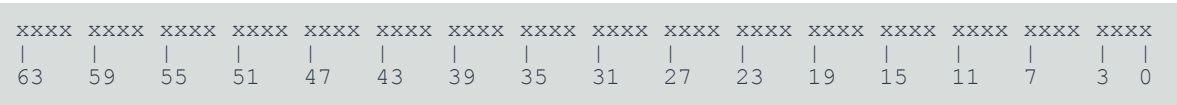
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-383: AARCH64_TRCCNTVR<n> bit assignments

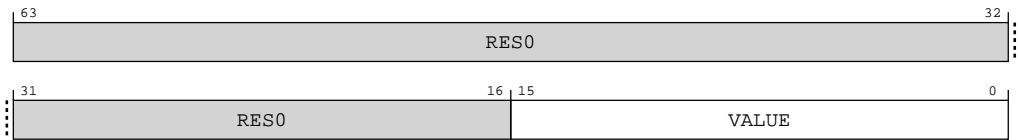


Table A-978: TRCCNTVR<n> bit descriptions

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] == 0b1.

MRS <Xt>, TRCCNTVR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'10':m[1:0]	0b101

MSR TRCCNTVR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	'10':m[1:0]	0b101

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 0b1.

MRS <Xt>, TRCCNTVR<m>

```
integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCNTVR[m];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCNTVR[m];
```

MSR TRCCNTVR<m>, <Xt>

```
integer m = UInt(CRm<1:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCNTVR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCNTVR[m] = X[t, 64];
```

A.2.8.5 TRCTRACEIDR, Trace ID Register

Sets the trace ID for instruction trace.

Configurations

AArch64 register TRCTRACEIDR bits [31:0] are architecturally mapped to External register [B.2.2.7.12 TRCTRACEIDR, Trace ID Register](#) on page 1996 bits [31:0].

Attributes

Width

64

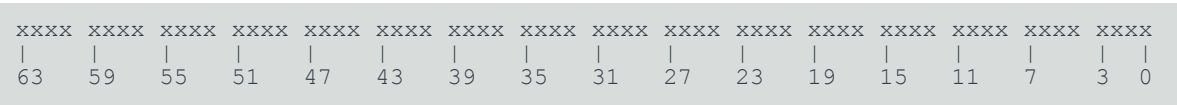
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-384: AARCH64_TRCTRACEIDR bit assignments

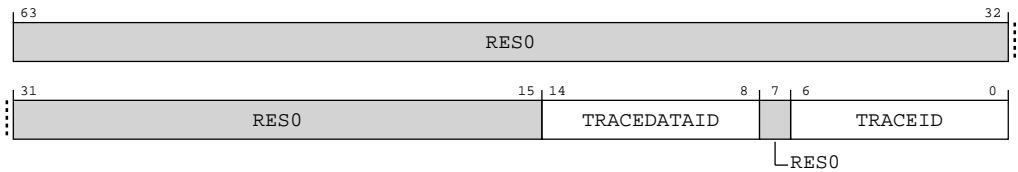


Table A-981: TRCTRACEIDR bit descriptions

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

Bits	Name	Description	Reset
[14:8]	TRACEDATAID	Data Trace ID. Sets the trace ID value for data trace. Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause UNPREDICTABLE behavior of the trace capture infrastructure. See the AMBA ATB Protocol Specification for information about which ATID values are reserved.	7{x}
[7]	RES0	Reserved	RES0
[6:0]	TRACEID	Trace ID. Sets the trace ID value for instruction trace. Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause UNPREDICTABLE behavior of the trace capture infrastructure. See the AMBA ATB Protocol Specification for information about which ATID values are reserved.	7{x}

Access

Must be programmed if implemented.

MRS <Xt>, TRCTRACEIDR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b001

MSR TRCTRACEIDR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b001

Accessibility

Must be programmed if implemented.

MRS <Xt>, TRCTRACEIDR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCTRACEIDR;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCTRACEIDR;

```


MSR TRCTRACEIDR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCTRACEIDR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCTRACEIDR = X[t, 64];
```

A.2.8.6 TRCVICTLR, ViewInst Main Control Register

Controls instruction trace filtering.

Configurations

AArch64 register TRCVICTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.13 TRCVICTLR, ViewInst Main Control Register](#) on page 1997 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-385: AARCH64_TRCVICTLR bit assignments

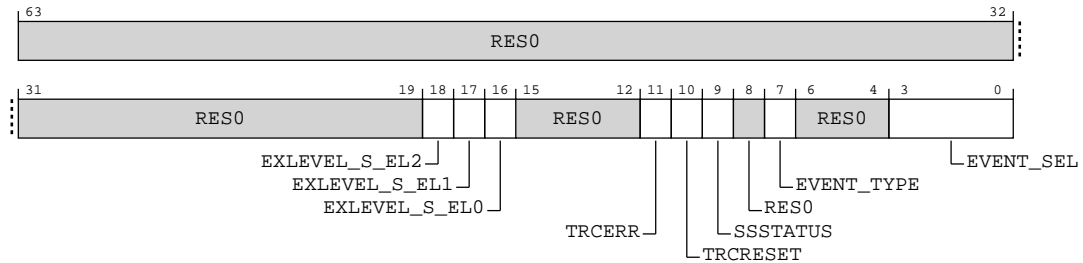


Table A-984: TRCVICTLR bit descriptions

Bits	Name	Description	Reset
[63:19]	RES0	Reserved	RES0
[18]	EXLEVEL_S_EL2	Filter instruction trace for EL2 in Secure state. 0b0 The trace unit generates instruction trace for EL2 in Secure state. 0b1 The trace unit does not generate instruction trace for EL2 in Secure state.	x
[17]	EXLEVEL_S_EL1	Filter instruction trace for EL1 in Secure state. 0b0 The trace unit generates instruction trace for EL1 in Secure state. 0b1 The trace unit does not generate instruction trace for EL1 in Secure state.	x
[16]	EXLEVEL_S_ELO	Filter instruction trace for ELO in Secure state. 0b0 The trace unit generates instruction trace for ELO in Secure state. 0b1 The trace unit does not generate instruction trace for ELO in Secure state.	x
[15:12]	RES0	Reserved	RES0
[11]	TRCERR	Controls the forced tracing of System Error exceptions. 0b0 Forced tracing of System Error exceptions is disabled. 0b1 Forced tracing of System Error exceptions is enabled.	x
[10]	TRCRESET	Controls the forced tracing of PE Resets. 0b0 Forced tracing of PE Resets is disabled. 0b1 Forced tracing of PE Resets is enabled.	x

Bits	Name	Description	Reset
[9]	SSSTATUS	<p>ViewInst start/stop function status.</p> <p>0b0</p> <p>Stopped State.</p> <p>The ViewInst start/stop function is in the stopped state.</p> <p>0b1</p> <p>Started State.</p> <p>The ViewInst start/stop function is in the started state.</p> <p>Before software enables the trace unit, it must write to this bit to set the initial state of the ViewInst start/stop function. If the ViewInst start/stop function is not used then set this bit to 0b1. Arm recommends that the value of this bit is set before each trace session begins.</p> <p>If the trace unit becomes disabled while a start point or stop point is still speculative, then the value of SSSTATUS is UNKNOWN and might represent the result of a speculative start point or stop point.</p> <p>If software which is running on the PE being traced disables the trace unit, either by clearing AArch64-TRCPRGCTLR.EN or locking the OS Lock, Arm recommends that a DSB and an ISB instruction are executed before disabling the trace unit to prevent any start points or stop points being speculative at the point of disabling the trace unit. This procedure assumes that all start points or stop points occur before the barrier instructions are executed. The procedure does not guarantee that there are no speculative start points or stop points when disabling, although it helps minimize the probability.</p>	x
[8]	RES0	Reserved	RES0
[7]	EVENT_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENT_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT_TYPE controls whether EVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx

Access

Must be programmed.

MRS <Xt>, TRCVICTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b010

MSR TRCVICTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b010

Accessibility

Must be programmed.

MRS <Xt>, TRCVICTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVICTLR;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVICTLR;

```

MSR TRCVICTLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVICTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVICTLR = X[t, 64];

```

A.2.8.7 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

AArch64 register TRCIDR8 bits [31:0] are architecturally mapped to External register [B.2.2.7.26 TRCIDR8, ID Register 8](#) on page 2024 bits [31:0].

Accessibility

MRS <Xt>, TRCIDR8

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR8;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR8;
```

A.2.8.8 TRCIMSPECO, IMP DEF Register 0

TRCIMSPECO shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

AArch64 register TRCIMSPECO bits [31:0] are architecturally mapped to External register [B.2.2.7.32 TRCIMSPECO, IMP DEF Register 0](#) on page 2030 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-387: AARCH64_TRCIMSPECO bit assignments

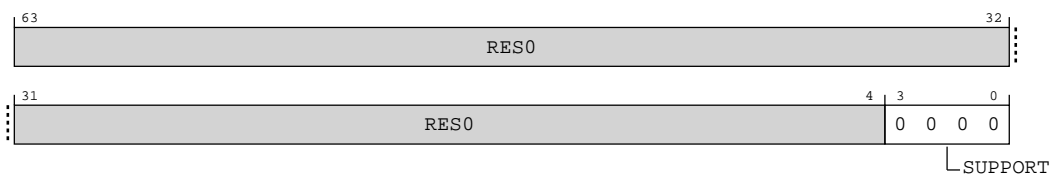


Table A-989: TRCIMSPECO bit descriptions

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>, TRCIMSPECO

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

MSR TRCIMSPECO, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, TRCIMSPECO

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIMSPECO;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIMSPECO;
```

MSR TRCIMSPECO, <Xt>

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

```
elseif PSTATE.EL == EL1 then
  if CPACR_EL1.TTA == '1' then
    AArch64.SystemAccessTrap(EL1, 0x18);
  elseif CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    TRCIMSPEC0 = X[t, 64];
elseif PSTATE.EL == EL2 then
  if CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    TRCIMSPEC0 = X[t, 64];
```

A.2.8.9 TRCPRGCTLR, Programming Control Register

Enables the trace unit.

Configurations

AArch64 register TRCPRGCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.1 TRCPRGCTLR, Programming Control Register](#) on page 1973 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-388: AARCH64_TRCPRGCTLR bit assignments

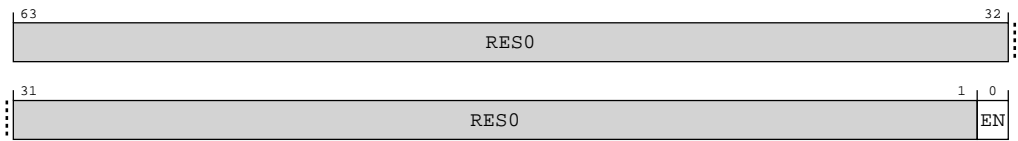


Table A-992: TRCPRGCTLR bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	EN	Trace unit enable. 0b0 The trace unit is disabled. 0b1 The trace unit is enabled.	0b0

Access

Must be programmed.

MRS <Xt>, TRCPRGCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b000

MSR TRCPRGCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b000

Accessibility

Must be programmed.

MRS <Xt>, TRCPRGCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCPRGCTLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCPRGCTLR;

```

MSR TRCPRGCTLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCPRGCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then

```

```
if CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    TRCPRGCTLR = X[t, 64];
```

A.2.8.10 TRCVIIECTLR, ViewInst Include/Exclude Control Register

Use this to select, or read, the Address Range Comparators for the ViewInst include/exclude function.

Configurations

AArch64 register TRCVIIECTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.14 TRCVIIECTLR, ViewInst Include/Exclude Control Register](#) on page 2000 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-389: AARCH64_TRCVIIECTLR bit assignments

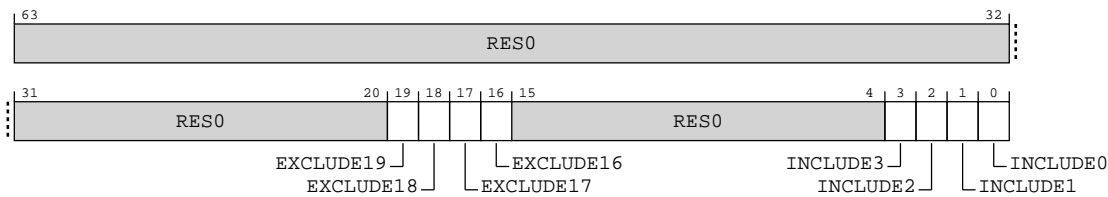


Table A-995: TRCVIIECTLR bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0
[19:16]	EXCLUDE<m>, bit[m], where m = 19 to 16	<p>Selects which Address Range Comparators are in use with the ViewInst exclude function.</p> <p>Each bit represents an Address Range Comparator, so bit[m] controls the selection of Address Range Comparator m.</p> <p>0b0</p> <p>The address range that Address Range Comparator m defines, is not selected for the ViewInst exclude function.</p> <p>0b1</p> <p>The address range that Address Range Comparator m defines, is selected for the ViewInst exclude function.</p>	xxxx
[15:4]	RES0	Reserved	RES0
[3:0]	INCLUDE<m>, bit[m], where m = 3 to 0	<p>Selects which Address Range Comparators are in use with the ViewInst include function.</p> <p>Each bit represents an Address Range Comparator, so bit[m] controls the selection of Address Range Comparator m.</p> <p>Selecting no comparators for the ViewInst include function indicates that all instructions are included by default.</p> <p>The ViewInst exclude function then indicates which ranges are excluded.</p> <p>0b0</p> <p>The address range that Address Range Comparator m defines, is not selected for the ViewInst include function.</p> <p>0b1</p> <p>The address range that Address Range Comparator m defines, is selected for the ViewInst include function.</p>	xxxx

Access

Must be programmed if AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

MRS <Xt>, TRCVIIECTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b010

MSR TRCVIIECTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b010

Accessibility

Must be programmed if AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

MRS <Xt>, TRCVIIECTLR

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

```
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
      AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      X[t, 64] = TRCVIIECTLR;
  elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      X[t, 64] = TRCVIIECTLR;
```

MSR TRCVIIECTLR, <Xt>

```
  if PSTATE.EL == EL0 then
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
      AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      TRCVIIECTLR = X[t, 64];
  elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      TRCVIIECTLR = X[t, 64];
```

A.2.8.11 TRCIDR9, ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR9 bits [31:0] are architecturally mapped to External register [B.2.2.7.27 TRCIDR9, ID Register 9](#) on page 2025 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000	0000	0000	0000	0000	0010	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-390: AARCH64_TRCIDR9 bit assignments

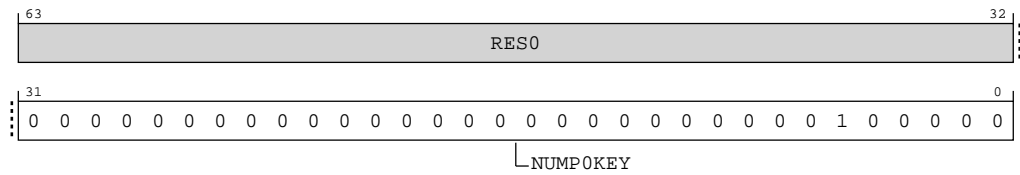


Table A-998: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	NUMPOKEY	Indicates the number of PO right-hand keys. 0b0000000000000000000000000100000 32 PO right-hand keys.	0x00000020

Access

MRS <Xt>, TRCIDR9

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0001	0b110

Accessibility

MRS <Xt>, TRCIDR9

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR9;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR9;
```

A.2.8.12 TRCVISSCTLR, ViewInst Start/Stop Control Register

Use this to select, or read, the Single Address Comparators for the ViewInst start/stop function.

Configurations

AArch64 register TRCVISSCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.15 TRCVISSCTLR, ViewInst Start/Stop Control Register](#) on page 2002 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-391: AARCH64_TRCVISSCTLR bit assignments

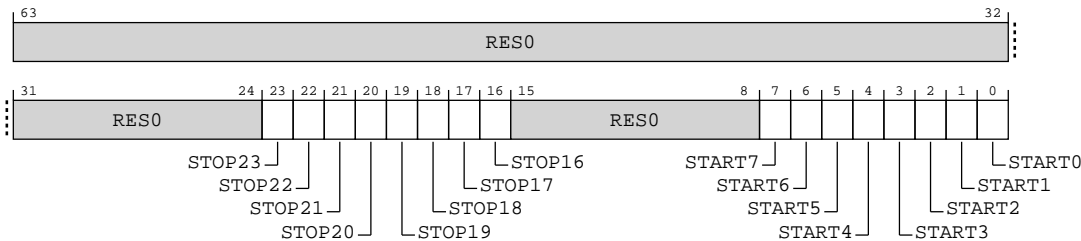


Table A-1000: TRCVISSCTLR bit descriptions

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

Bits	Name	Description	Reset
[23:16]	STOP<m>, bit[m], where m = 23 to 16	<p>Selects which Single Address Comparators are in use with ViewInst start/stop function, for the purpose of stopping trace.</p> <p>0b0 The Single Address Comparator m, is not selected as a stop resource.</p> <p>0b1 The Single Address Comparator m, is selected as a stop resource.</p>	8 {x}
[15:8]	RES0	Reserved	RES0
[7:0]	START<m>, bit[m], where m = 7 to 0	<p>Selects which Single Address Comparators are in use with ViewInst start/stop function, for the purpose of starting trace.</p> <p>0b0 The Single Address Comparator m, is not selected as a start resource.</p> <p>0b1 The Single Address Comparator m, is selected as a start resource.</p>	8 {x}

Access

Must be programmed if AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

For any 2 comparators selected for the ViewInst start/stop function, the comparator containing the lower address must be a lower numbered comparator.

MRS <Xt>, TRCVISSCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b010

MSR TRCVISSCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0010	0b010

Accessibility

Must be programmed if AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

For any 2 comparators selected for the ViewInst start/stop function, the comparator containing the lower address must be a lower numbered comparator.

MRS <Xt>, TRCVISSCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVISSCTLR;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);

```

```
else
    X[t, 64] = TRCVISSCTLR;
```

MSR TRCVISSCTLR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVISSCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVISSCTLR = X[t, 64];
```

A.2.8.13 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR10 bits [31:0] are architecturally mapped to External register [B.2.2.7.28 TRCIDR10, ID Register 10](#) on page 2026 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000	0000	0000	0000	0000	0000	1000
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.

Attributes

Width

64

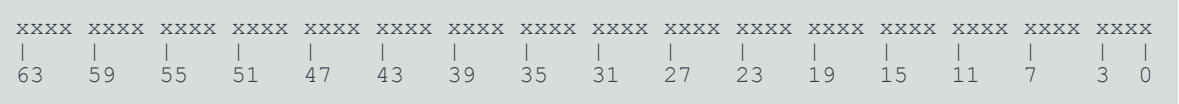
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-393: AARCH64_TRCSTATR bit assignments

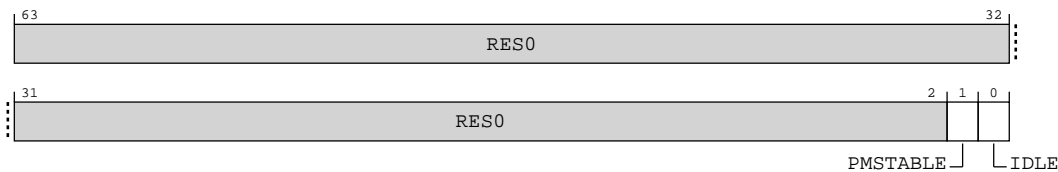


Table A-1005: TRCSTATR bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	RES0
[1]	PMSTABLE	Programmers' model stable. 0b0 The programmers' model is not stable. 0b1 The programmers' model is stable. This bit is UNKNOWN while the trace unit is enabled.	x
[0]	IDLE	Idle status. 0b0 The trace unit is not idle. 0b1 The trace unit is idle.	x

Access

MRS <Xt>, TRCSTATR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b000

Accessibility

MRS <Xt>, TRCSTATR

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSTATR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSTATR;
```

A.2.8.15 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR11 bits [31:0] are architecturally mapped to External register [B.2.2.7.29 TRCIDR11, ID Register 11](#) on page 2027 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-394: AARCH64_TRCIDR11 bit assignments

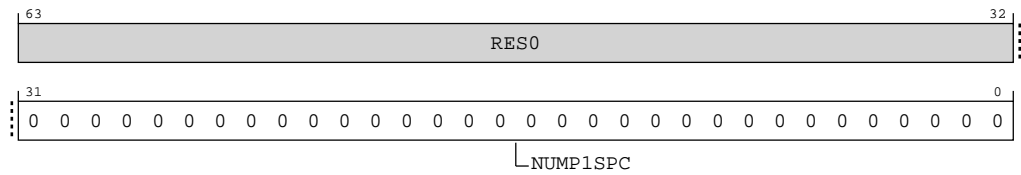


Table A-1007: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	NUMP1SPC	Indicates the number of special P1 right-hand keys. 0b00000000000000000000000000000000 No special P1 right-hand keys.	0x00000000

Access

MRS <Xt>, TRCIDR11

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b110

Accessibility

MRS <Xt>, TRCIDR11

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR11;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR11;
```

A.2.8.16 TRCCONFIGR, Trace Configuration Register

Controls the tracing options.

Configurations

AArch64 register TRCCONFIGR bits [31:0] are architecturally mapped to External register [B.2.2.7.3 TRCCONFIGR, Trace Configuration Register](#) on page 1976 bits [31:0].

Attributes

Width

64

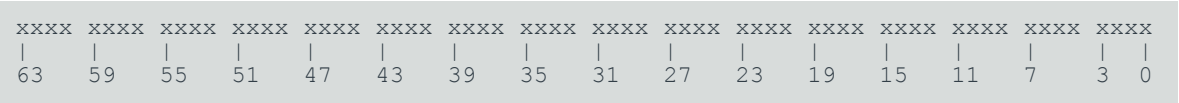
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-395: AARCH64_TRCCONFIGR bit assignments

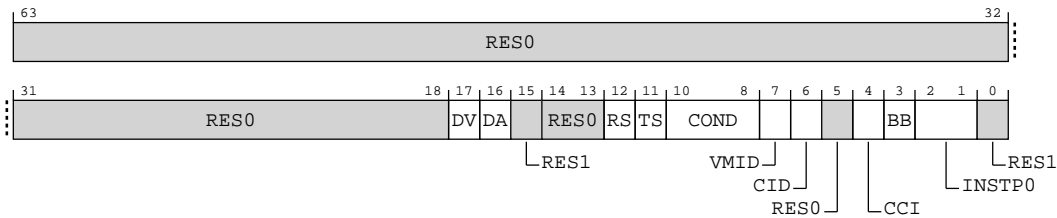


Table A-1009: TRCCONFIGR bit descriptions

Bits	Name	Description	Reset
[63:18]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[17]	DV	Data value tracing bit. 0b0 Data value tracing is disabled. 0b1 Data value tracing is enabled when INSTP0 is not 0b00.	x
[16]	DA	Data address tracing bit. 0b0 Data address tracing is disabled. 0b1 Data address tracing is enabled when INSTP0 is not 0b00.	x
[15]	RES1	Reserved	RES1
[14:13]	RES0	Reserved	RES0
[12]	RS	Return stack control. 0b0 Return stack is disabled. 0b1 Return stack is enabled.	x
[11]	TS	Global timestamp tracing control. 0b0 Global timestamp tracing is disabled. 0b1 Global timestamp tracing is enabled.	x
[10:8]	COND	Conditional instruction tracing bit. The permitted values are: 0b000 Conditional instruction tracing is disabled. 0b001 Conditional load instructions are traced. 0b010 Conditional store instructions are traced. 0b011 Conditional load and store instructions are traced. 0b111 All conditional instructions are traced.	xxx
[7]	VMID	Virtual context identifier tracing control. 0b0 Virtual context identifier tracing is disabled. 0b1 Virtual context identifier tracing is enabled.	x

Bits	Name	Description	Reset
[6]	CID	Context identifier tracing control. 0b0 Context identifier tracing is disabled. 0b1 Context identifier tracing is enabled.	x
[5]	RES0	Reserved	RES0
[4]	CCI	Cycle counting instruction tracing control. 0b0 Cycle counting instruction tracing is disabled. 0b1 Cycle counting instruction tracing is enabled.	x
[3]	BB	Branch broadcasting control. 0b0 Branch broadcasting is disabled. 0b1 Branch broadcasting is enabled.	x
[2:1]	INSTPO	Instruction P0 field. This field controls whether load and store instructions are traced as P0 instructions: 0b00 Do not trace load and store instructions as P0 instructions. 0b01 Trace load instructions as P0 instructions. 0b10 Trace store instructions as P0 instructions. 0b11 Trace load and store instructions as P0 instructions.	xx
[0]	RES1	Reserved	RES1

Access

Must always be programmed.

TRCCONFIGR.QE must be set to 0b00 if TRCCONFIGR.BB is not 0b0.

MRS <Xt>, TRCCONFIGR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b000

MSR TRCCONFIGR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0100	0b000

Accessibility

Must always be programmed.

TRCCONFIGR.QE must be set to 0b00 if TRCCONFIGR.BB is not 0b0.

MRS <Xt>, TRCCONFIGR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCONFIGR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCONFIGR;

```

MSR TRCCONFIGR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCONFIGR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCONFIGR = X[t, 64];

```

A.2.8.17 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR12 bits [31:0] are architecturally mapped to External register [B.2.2.7.30 TRCIDR12, ID Register 12](#) on page 2028 bits [31:0].

Attributes

Width

64

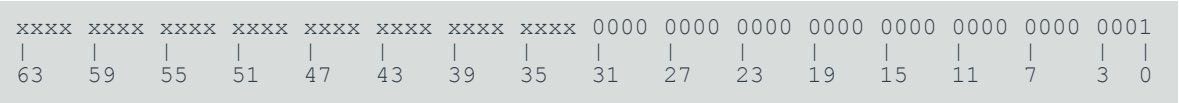
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-396: AArch64_TRCIDR12 bit assignments

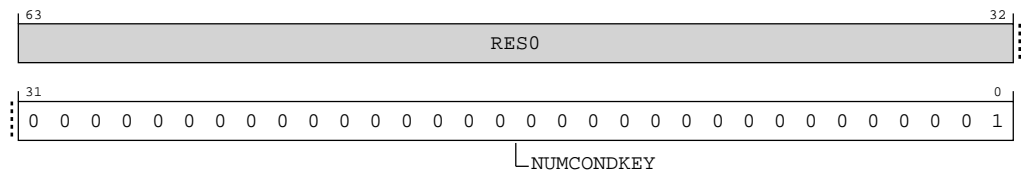


Table A-1012: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	NUMCONDKEY	Indicates the number of conditional instruction right-hand keys. 0b00000000000000000000000000000001 1 conditional instruction right-hand key.	0x00000001

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 CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR12;
```

```
elseif PSTATE.EL == EL2 then
  if CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    X[t, 64] = TRCIDR12;
```

A.2.8.18 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR13 bits [31:0] are architecturally mapped to External register [B.2.2.7.31 TRCIDR13, ID Register 13](#) on page 2029 bits [31:0].

Attributes

Width

64

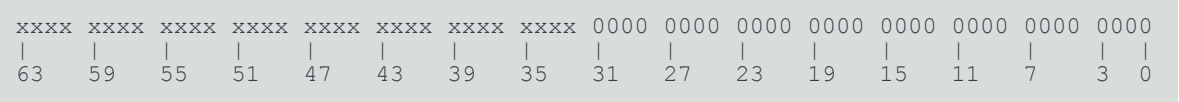
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-397: AARCH64_TRCIDR13 bit assignments

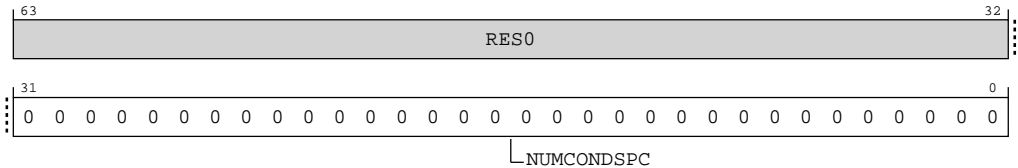


Table A-1014: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:0]	NUMCONDSPC	Indicates the number of special conditional instruction right-hand keys. 0b00000000000000000000000000000000 No special conditional instruction right-hand keys.	0x00000000

Access
MRS <Xt>, TRCIDR13

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0101	0b110

Accessibility
MRS <Xt>, TRCIDR13

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR13;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR13;
```

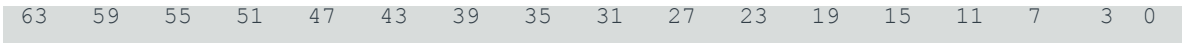
A.2.8.19 TRCAUXCTLR, Auxillary Control Register

Trace micro-architectural control bits.

Configurations
AArch64 register TRCAUXCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.4 TRCAUXCTLR, Auxillary Control Register](#) on page 1979 bits [31:0].

Attributes
Width
64
Functional group
Trace unit registers
Access type
See bit descriptions
Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-398: AARCH64_TRCAUXCTLR bit assignments

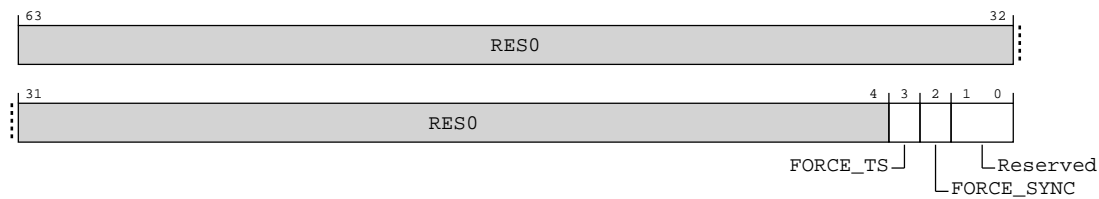


Table A-1016: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3]	FORCE_TS	Force overflow when no timestamp is generated between two trace info packets. 0b0 Force overflow disabled. 0b1 Force overflow enabled.	0b0
[2]	FORCE_SYNC	Force overflow when the generation of trace synchronisation packets is delayed. 0b0 Force overflow disabled. 0b1 Force overflow enabled.	0b0
[1:0]	Reserved	Trace unit control options which are reserved for Arm internal use. 0b00 Any trace unit control options affected by this register are disabled. All other values are reserved for Arm internal use. Arm strongly recommends that you do not modify this field unless directed by Arm.	0b00

Access

If this register is set to nonzero then it might cause the behavior of a trace unit to contradict the architecture specification. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

MRS <Xt>, TRCAUXCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b000

MSR TRCAUXCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b000

Accessibility

If this register is set to nonzero then it might cause the behavior of a trace unit to contradict the architecture specification. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

MRS <Xt>, TRCAUXCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCAUXCTLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCAUXCTLR;

```

MSR TRCAUXCTLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCAUXCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCAUXCTLR = X[t, 64];

```

A.2.8.20 TRCSEQRSTEV, Sequencer Reset Control Register

Moves the sequencer to state 0 when a programmed resource event occurs.

Configurations

AArch64 register TRCSEQRSTEV bits [31:0] are architecturally mapped to External register

[B.2.2.7.20 TRCSEQRSTEV, Sequencer Reset Control Register](#) on page 2014 bits [31:0].

Attributes

Width

64

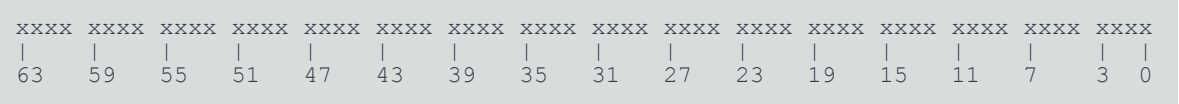
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-399: AARCH64_TRCSEQRSTEVr bit assignments

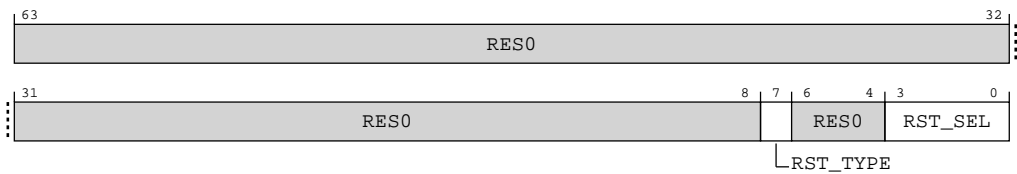


Table A-1019: TRCSEQRSTEVr bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7]	RST_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. RST_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. RST_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. RST_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:0]	RST_SEL	Defines the selected Resource Selector or pair of Resource Selectors. RST_TYPE controls whether RST_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire.	xxxx

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

MRS <Xt>, TRCSEQRSTEV

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b100

MSR TRCSEQRSTEV, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0110	0b100

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

MRS <Xt>, TRCSEQRSTEV

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSEQRSTEV;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSEQRSTEV;

```

MSR TRCSEQRSTEV, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSEQRSTEV = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);

```

```
else
    TRCSEQRSTEVr = X[t, 64];
```

A.2.8.21 TRCSEQSTR, Sequencer State Register

Use this to set, or read, the sequencer state.

Configurations

AArch64 register TRCSEQSTR bits [31:0] are architecturally mapped to External register [B.2.2.7.21 TRCSEQSTR, Sequencer State Register](#) on page 2015 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-400: AARCH64_TRCSEQSTR bit assignments

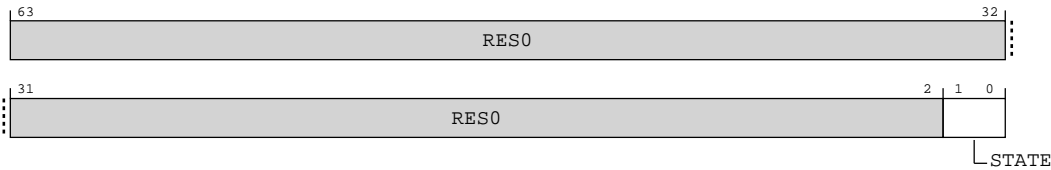


Table A-1022: TRCSEQSTR bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1:0]	STATE	Set or returns the state of the sequencer. 0b00 State 0. 0b01 State 1. 0b10 State 2. 0b11 State 3.	xx

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

MRS <Xt>, TRCSEQSTR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0111	0b100

MSR TRCSEQSTR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0111	0b100

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

MRS <Xt>, TRCSEQSTR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSEQSTR;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSEQSTR;

```

MSR TRCSEQSTR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then

```

```
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSEQSTR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSEQSTR = X[t, 64];
```

A.2.8.22 TRCEVENTCTLOR, Event Control 0 Register

Controls the generation of tracing events.

Configurations

AArch64 register TRCEVENTCTLOR bits [31:0] are architecturally mapped to External register [B.2.2.7.5 TRCEVENTCTLOR, Event Control 0 Register](#) on page 1981 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-401: AARCH64_TRCEVENTCTL0R bit assignments

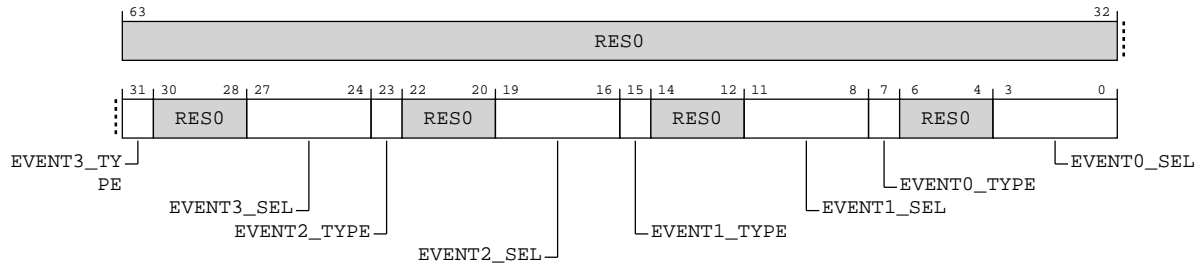


Table A-1025: TRCEVENTCTL0R bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	EVENT3_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT3_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT3_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT3_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[30:28]	RES0	Reserved	RES0
[27:24]	EVENT3_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT3_TYPE controls whether EVENT3_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p> <p>When any of the selected resource events occurs and AArch64-TRCEVENTCTL1R.INSTEN[3] == 0b1, then Event element 3 is generated in the instruction trace element stream.</p>	xxxx
[23]	EVENT2_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT2_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT2_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT2_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x

Bits	Name	Description	Reset
[22:20]	RES0	Reserved	RES0
[19:16]	EVENT2_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT2_TYPE controls whether EVENT2_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p> <p>When any of the selected resource events occurs and AArch64-TRCEVENTCTL1R.INSTEN[2] == 0b1, then Event element 3 is generated in the instruction trace element stream.</p>	xxxx
[15]	EVENT1_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT1_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT1_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT1_SEL[3] is RES0.</p>	x
[14:12]	RES0	Reserved	RES0
[11:8]	EVENT1_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT1_TYPE controls whether EVENT1_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p> <p>When any of the selected resource events occurs and AArch64-TRCEVENTCTL1R.INSTEN[1] == 0b1, then Event element 3 is generated in the instruction trace element stream.</p>	xxxx
[7]	EVENT0_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT0_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT0_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT0_SEL[3] is RES0.</p>	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENT0_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT0_TYPE controls whether EVENT0_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p> <p>When any of the selected resource events occurs and AArch64-TRCEVENTCTL1R.INSTEN[0] == 0b1, then Event element 3 is generated in the instruction trace element stream.</p>	xxxx

Access

Must be programmed if implemented.

MRS <Xt>, TRCEVENTCTLOR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b000

MSR TRCEVENTCTLOR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b000

Accessibility

Must be programmed if implemented.

MRS <Xt>, TRCEVENTCTLOR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCEVENTCTLOR;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCEVENTCTLOR;

```

MSR TRCEVENTCTLOR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCEVENTCTLOR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCEVENTCTLOR = X[t, 64];

```

A.2.8.23 TRCVDCTLR, ViewData Main Control Register

Controls data trace filtering.

Configurations

AArch64 register TRCVDCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.16 TRCVDCTLR, ViewData Main Control Register](#) on page 2004 bits [31:0].

Attributes

Width

64

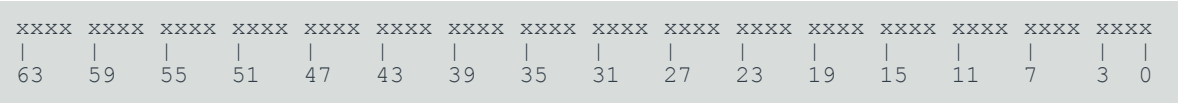
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-402: AARCH64_TRCVDCTLR bit assignments

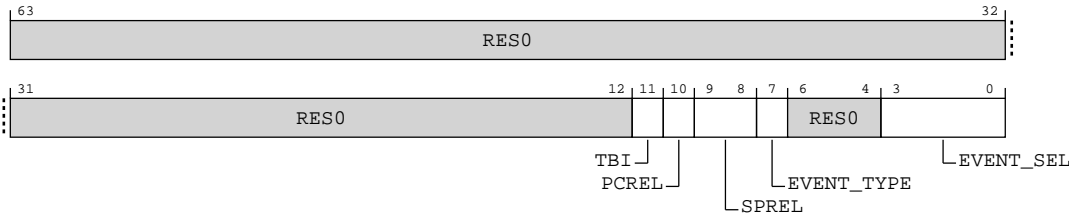


Table A-1028: TRCVDCTLR bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[11]	TBI	Controls which information a trace unit populates in bits[63:56] of the data address. 0b0 The trace unit assigns bits[63:56] to have the same value as bit[55] of the data address, that is, it sign-extends the value. 0b1 The trace unit assigns bits[63:56] to have the same value as bits[63:56] of the data address.	x
[10]	PCREL	Controls whether a trace unit traces data for transfers that are relative to the Program Counter (PC). 0b0 The trace unit does not affect the tracing of PC-relative transfers. 0b1 The trace unit does not trace the address or value portions of PC-relative transfers. This bit only affects PC-relative transfers that use the PC as a base register plus an immediate offset.	x
[9:8]	SPREL	Controls whether a trace unit traces data for transfers that are relative to the Stack Pointer (SP). 0b00 The trace unit does not affect the tracing of SP-relative transfers. 0b01 Reserved. 0b10 The trace unit does not trace the address portion of SP-relative transfers. If data value tracing is enabled then the trace unit generates a P1 data address element. 0b11 The trace unit does not trace the address or value portions of SP-relative transfers.	xx
[7]	EVENT_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. EVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. EVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. EVENT_TYPE controls whether EVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire.	xxxx

Access

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when data tracing is enabled, that is, when either AArch64-TRCCONFIGR.DA==1 or AArch64-TRCCONFIGR.DV==1.

MRS <Xt>, TRCVDCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b010

MSR TRCVDCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b010

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when data tracing is enabled, that is, when either AArch64-TRCCONFIGR.DA==1 or AArch64-TRCCONFIGR.DV==1.

MRS <Xt>, TRCVDCTLR

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVDCTLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVDCTLR;
```

MSR TRCVDCTLR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVDCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVDCTLR = X[t, 64];
```


A.2.8.24 TRCEXTINSELR, External Input Select Register

Use this to set, or read, which external inputs are resources to the trace unit.

Configurations

AArch64 register TRCEXTINSELR bits [31:0] are architecturally mapped to External register [B.2.2.7.22 TRCEXTINSELR, External Input Select Register](#) on page 2017 bits [31:0].

Attributes

Width

64

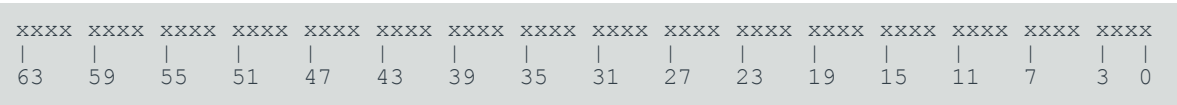
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-403: AARCH64_TRCEXTINSELR bit assignments

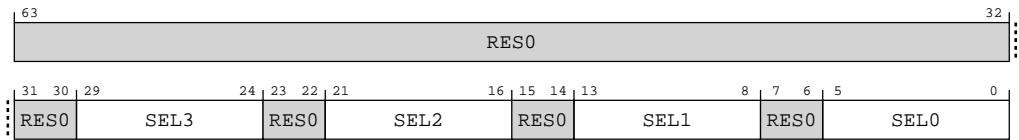


Table A-1031: TRCEXTINSELR bit descriptions

Bits	Name	Description	Reset
[63:30]	RES0	Reserved	RES0
[29:24]	SEL3	Selects external input resource #3.	6 { x }
[23:22]	RES0	Reserved	RES0
[21:16]	SEL2	Selects external input resource #2.	6 { x }
[15:14]	RES0	Reserved	RES0
[13:8]	SEL1	Selects external input resource #1.	6 { x }
[7:6]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[5:0]	SELO	Selects external input resource #0.	6{x}

Access

Might ignore writes when the trace unit is enabled or not idle.

MRS <Xt>, TRCEXTINSEL

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b100

MSR TRCEXTINSEL, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b100

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

MRS <Xt>, TRCEXTINSEL

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCEXTINSEL;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCEXTINSEL;

```

MSR TRCEXTINSEL, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCEXTINSEL = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCEXTINSEL = X[t, 64];

```

A.2.8.25 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR0 bits [31:0] are architecturally mapped to External register [B.2.2.7.33 TRCIDR0, ID Register 0](#) on page 2031 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx00	1000	xxxx	xx00	0x00	111x	1111	111x
63	59	55	51	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-404: AARCH64_TRCIDR0 bit assignments

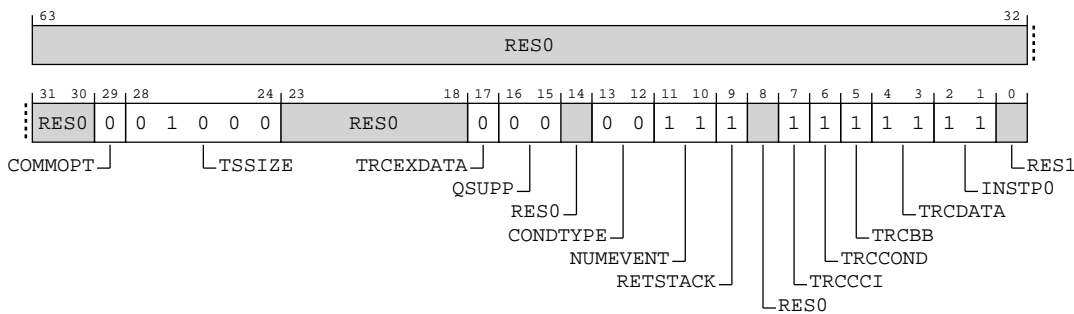


Table A-1034: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:30]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[29]	COMMOPT	Indicates how the commit field in Cycle count packets is interpreted. 0b0 Commit mode 0. Cycle count packets includes the Commit element to which the Cycle Count element is attached.	0b0
[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
[23:18]	RES0	Reserved	RES0
[17]	TRCEXDATA	Indicates if the trace unit implements tracing of data transfers for exceptions and exception returns. 0b0 Tracing of data transfers for exceptions and exception returns not implemented.	0b0
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. 0b00 Q element support is not implemented.	0b00
[14]	RES0	Reserved	RES0
[13:12]	CONDTYPE	Indicates how conditional instructions are traced. 0b00 Conditional instructions are traced with an indication of whether they pass or fail their condition code check.	0b00
[11:10]	NUMEVENT	Indicates the number of trace events implemented. 0b11 The trace unit supports 4 trace events.	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. 0b1 Conditional instruction tracing implemented.	0b1
[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. 0b11 Data tracing implemented.	0b11
[2:1]	INSTPO	Indicates if load and store instructions are PO instructions. 0b11 Load and store instructions are PO instructions.	0b11

Bits	Name	Description	Reset
[0]	RES1	Reserved	RES1

Access
MRS <Xt>, TRCIDR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1000	0b111

Accessibility
MRS <Xt>, TRCIDR0

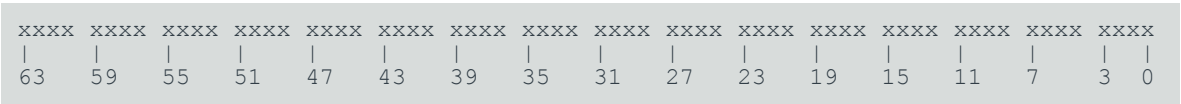
```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR0;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR0;
```

A.2.8.26 TRCEVENTCTL1R, Event Control 1 Register

Controls the behavior of the tracing events that AArch64-TRCEVENTCTL0R selects.

Configurations
AArch64 register TRCEVENTCTL1R bits [31:0] are architecturally mapped to External register [B.2.2.7.6 TRCEVENTCTL1R, Event Control 1 Register](#) on page 1984 bits [31:0].

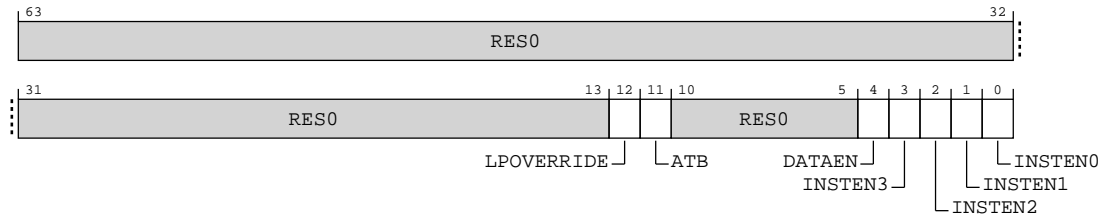
Attributes
Width
64
Functional group
Trace unit registers
Access type
See bit descriptions
Reset value



**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-405: AARCH64_TRCEVENTCTL1R bit assignments**Table A-1036: TRCEVENTCTL1R bit descriptions**

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	RES0
[12]	LPOVERRIDE	<p>Low-power Override Mode select.</p> <p>0b0</p> <p>Trace unit Low-power Override Mode is not enabled. That is, the trace unit is permitted to enter low-power state.</p> <p>0b1</p> <p>Trace unit Low-power Override Mode is enabled. That is, entry to a low-power state does not affect the trace unit resources or trace generation.</p>	x
[11]	ATB	<p>AMBA Trace Bus (ATB) trigger enable.</p> <p>When trace event 0 occurs the trace unit sets:</p> <ul style="list-style-type: none"> • ATID == 0x7D. • ATDATA to the value of AArch64-TRCTRACEIDR. <p>If the width of ATDATA is greater than the width of AArch64-TRCTRACEIDR.TRACEID then the trace unit zeros the upper ATDATA bits.</p> <p>If trace event 0 is programmed to occur based on program execution, such as an address comparator, the ATB trigger might not be inserted into the ATB stream at the same time as any trace generated by that program execution is output by the trace unit. Typically, the generated trace might be buffered in a trace unit which means that the ATB trigger would be output before the associated trace is output.</p> <p>If trace event 0 is asserted multiple times in close succession, the trace unit is required to generate an ATB trigger for the first assertion, but might ignore one or more of the subsequent assertions. Arm recommends that the window in which ETMEvent 0 is ignored is limited only by the time taken to output an ATB trigger.</p> <p>0b0</p> <p>ATB trigger is disabled.</p> <p>0b1</p> <p>ATB trigger is enabled.</p>	x

Bits	Name	Description	Reset
[10:5]	RES0	Reserved	RES0
[4]	DATAEN	Data event enable bit. 0b0 If event 0 occurs, the trace unit does not generate an Event element. 0b1 If event 0 occurs, the trace unit generates an Event element in the data trace stream.	x
[3:0]	INSTEN<m>, bit[m], where m = 3 to 0	Event element control. 0b0 The trace unit does not generate an Event element m. 0b1 The trace unit generates an Event element m.	xxxx

Access

Must be programmed.

MRS <Xt>, TRCEVENTCTL1R

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b000

MSR TRCEVENTCTL1R, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b000

Accessibility

Must be programmed.

MRS <Xt>, TRCEVENTCTL1R

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCEVENTCTL1R;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCEVENTCTL1R;

```

MSR TRCEVENTCTL1R, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```
if CPACR_EL1.TTA == '1' then
    AArch64.SystemAccessTrap(EL1, 0x18);
elsif CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    TRCEVENTCTL1R = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCEVENTCTL1R = X[t, 64];
```

A.2.8.27 TRCVDSACCTLR, ViewData Include/Exclude Single Address Comparator Control Register

Use this to select, or read, the Single Address Comparators for the ViewData include/exclude control.

Configurations

AArch64 register TRCVDSACCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.17 TRCVDSACCTLR, ViewData Include/Exclude Single Address Comparator Control Register](#) on page 2006 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-406: AARCH64_TRCVDSACCTLR bit assignments

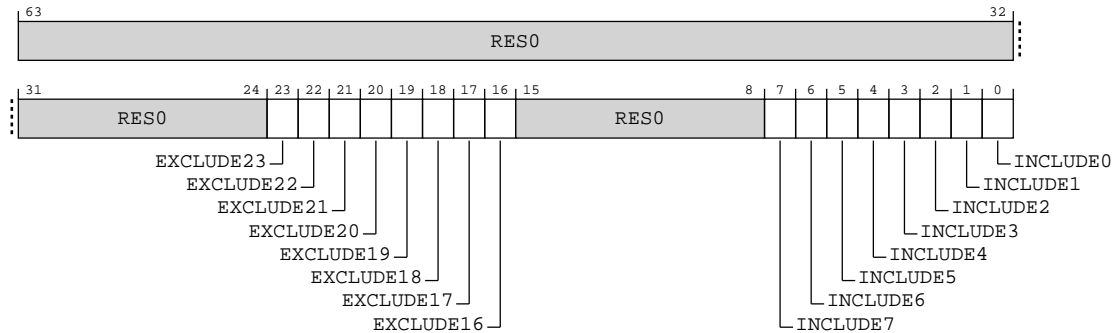


Table A-1039: TRCVDSACCTLR bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	RES0
[23:16]	EXCLUDE<m>, bit[m], where $m = 23$ to 16	<p>Selects which single address comparators are in use with ViewData exclude control..</p> <p>0b0</p> <p>The single address comparator m, is not selected for exclude control.</p> <p>0b1</p> <p>The single address comparator m, is selected for exclude control.</p>	$8\{x\}$
[15:8]	RES0	Reserved	RES0
[7:0]	INCLUDE<m>, bit[m], where $m = 7$ to 0	<p>Selects which single address comparators are in use with ViewData include control.</p> <p>0b0</p> <p>The single address comparator m, is not selected for include control.</p> <p>0b1</p> <p>The single address comparator m, is selected for include control.</p> <p>If no single address comparators and no address range comparators are selected to be included, then all data transfers are included by default. The exclude control then indicates which data transfers are excluded.</p>	$8\{x\}$

Access

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, AArch64-TRCCONFIGR.DA==1 or AArch64-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects a single address comparator that is either:

- Programmed to be sensitive to a data value comparator.

- Programmed to be an instruction address comparator.

In these situations, data transfers might be traced unexpectedly or might not be traced.

MRS <Xt>, TRCVDSACCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b010

MSR TRCVDSACCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1001	0b010

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, AArch64-TRCCONFIGR.DA==1 or AArch64-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects a single address comparator that is either:

- Programmed to be sensitive to a data value comparator.
- Programmed to be an instruction address comparator.

In these situations, data transfers might be traced unexpectedly or might not be traced.

MRS <Xt>, TRCVDSACCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVDSACCTLR;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVDSACCTLR;

```

MSR TRCVDSACCTLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then

```

```
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVDSACCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVDSACCTLR = X[t, 64];
```

A.2.8.28 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR1 bits [31:0] are architecturally mapped to External register [B.2.2.7.34 TRCIDR1, ID Register 1](#) on page 2033 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0100	0001	xxxx	xxxx	xxxx	0100	0101	0111
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-407: AARCH64_TRCIDR1 bit assignments

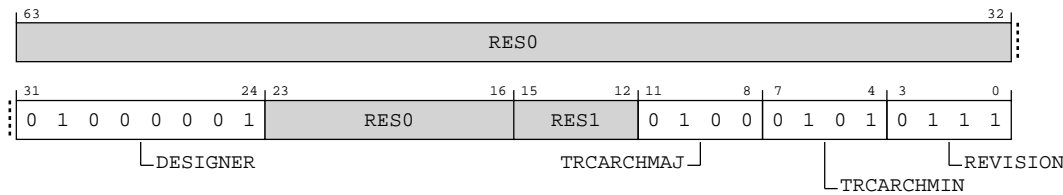


Table A-1042: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:24]	DESIGNER	Indicates which company designed the trace unit. 0b01000001 Arm Limited.	0x41
[23:16]	RES0	Reserved	RES0
[15:12]	RES1	Reserved	RES1
[11:8]	TRCARCHMAJ	Major architecture version. 0b0100 ETMv4.	0b0100
[7:4]	TRCARCHMIN	Minor architecture version. 0b0101 ETMv4.5.	0b0101
[3:0]	REVISION	Implementation revision. 0b0111 Revision 7.	0b0111

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 CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR1;
    endif
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR1;
    endif
endif

```

A.2.8.29 TRCVDARCCTLR, ViewData Include/Exclude Address Range Comparator Control Register

Use this to select, or read, the Address Range Comparator for the ViewData include/exclude control.

Configurations

AArch64 register TRCVDARCCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.18 TRCVDARCCTLR, ViewData Include/Exclude Address Range Comparator Control Register](#) on page 2009 bits [31:0].

Attributes

Width

64

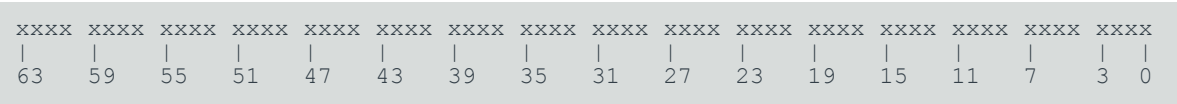
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-408: AARCH64_TRCVDARCCTLR bit assignments

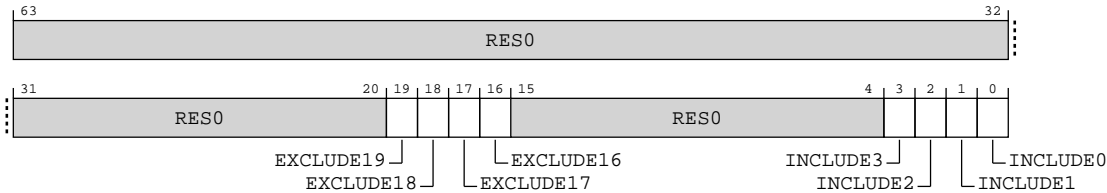


Table A-1044: TRCVDARCCTLR bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[19:16]	EXCLUDE<m>, bit[m], where m = 19 to 16	<p>Exclude range field. Selects which address range comparator pairs are in use with ViewData exclude control.</p> <p>0b0</p> <p>The address range that address range comparator pair m defines, is not selected for ViewData exclude control.</p> <p>0b1</p> <p>The address range that address range comparator pair m defines, is selected for ViewData exclude control.</p>	xxxx
[15:4]	RES0	Reserved	RES0
[3:0]	INCLUDE<m>, bit[m], where m = 3 to 0	<p>Include range field. Selects which address range comparator pairs are in use with ViewData include control.</p> <p>0b0</p> <p>The address range that address range comparator pair m defines, is not selected for ViewData include control.</p> <p>0b1</p> <p>The address range that address range comparator pair m defines, is selected for ViewData include control.</p> <p>If no single address comparators and no address range comparators are selected to be included, then all data transfers are included by default. The exclude control then indicates which data transfers are excluded.</p>	xxxx

Access

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, AArch64-TRCCONFIGR.DA==1 or AArch64-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects an address range comparator that is programmed to be sensitive to a data value comparator, data transfers might be traced unexpectedly or might not be traced.

MRS <Xt>, TRCVDARCCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b010

MSR TRCVDARCCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1010	0b010

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, AArch64-TRCCONFIGR.DA==1 or AArch64-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, AArch64-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects an address range comparator that is programmed to be sensitive to a data value comparator, data transfers might be traced unexpectedly or might not be traced.

MRS <Xt>, TRCVDARCCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVDARCCTLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVDARCCTLR;

```

MSR TRCVDARCCTLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVDARCCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVDARCCTLR = X[t, 64];

```

A.2.8.30 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR2 bits [31:0] are architecturally mapped to External register [B.2.2.7.35 TRCIDR2, ID Register 2](#) on page 2035 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1100	0000	1000	0100	0001	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-409: AARCH64_TRCIDR2 bit assignments

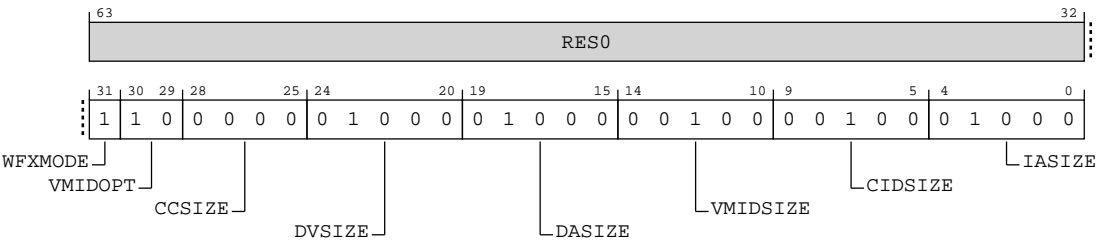


Table A-1047: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as PO instructions: 0b1 WFI and WFE instructions are classified as PO instructions.	0b1
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. 0b10 Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is RES1 .	0b10
[28:25]	CCSIZE	Indicates the size of the cycle counter. 0b0000 The cycle counter is 12 bits in length. All other values are reserved.	0b0000

Bits	Name	Description	Reset
[24:20]	DVSIZE	Indicates the data value size in bytes. 0b01000 Data value tracing has a maximum of 64-bit data values.	0b01000
[19:15]	DASIZE	Indicates the data address size in bytes. 0b01000 Data address tracing has a maximum of 64-bit data addresses.	0b01000
[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;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR2;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR2;

```

A.2.8.31 TRCSTALLCTLR, Stall Control Register

Enables trace unit functionality that prevents trace unit buffer overflows.

Configurations

AArch64 register TRCSTALLCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.7 TRCSTALLCTLR, Stall Control Register](#) on page 1986 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-410: AARCH64_TRCSTALLCTLR bit assignments

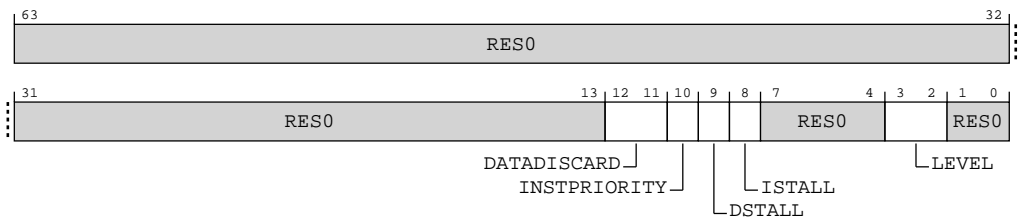


Table A-1049: TRCSTALLCTLR bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	RES0
[12:11]	DATADISCARD	Data discard field. Controls if a trace unit can discard data trace elements when the data trace buffer space is less than LEVEL. 0b00 The trace unit must not discard any data trace elements. 0b01 The trace unit can discard P1 and P2 elements associated with data loads. 0b10 The trace unit can discard P1 and P2 elements associated with data stores. 0b11 The trace unit can discard P1 and P2 elements associated with both data loads and stores.	xx

Bits	Name	Description	Reset
[10]	INSTPRIORITY	<p>Prioritize instruction trace bit. Controls if a trace unit can prioritize instruction trace when the instruction trace buffer space is less than LEVEL.</p> <p>0b0 The trace unit must not prioritize instruction trace.</p> <p>0b1 The trace unit can prioritize instruction trace. A trace unit might prioritize</p>	x
[9]	DSTALL	<p>Data stall bit. Controls if a trace unit can stall the PE when the data trace buffer space is less than LEVEL.</p> <p>0b0 The trace unit must not stall the PE.</p> <p>0b1 The trace unit can stall the PE.</p>	x
[8]	ISTALL	<p>Instruction stall control. Controls if a trace unit can stall the PE when the trace buffer space is less than LEVEL.</p> <p>0b0 The trace unit must not stall the PE.</p> <p>0b1 The trace unit can stall the PE.</p>	x
[7:4]	RES0	Reserved	RES0
[3:2]	LEVEL	<p>Threshold level field. The field supports 4 monotonic levels from 0b00 to 0b11.</p> <p>0b00 Minimal invasion. This setting has a greater risk of a trace unit buffer overflow.</p> <p>0b01 Small invasion.</p> <p>0b10 Big invasion.</p> <p>0b11 Maximum invasion. Reduced risk of a trace unit buffer overflow.</p>	xx
[1:0]	RES0	Reserved	RES0

Access

Must be programmed if implemented.

MRS <Xt>, TRCSTALLCTL

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b000

MSR TRCSTALLCTL, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1011	0b000

Accessibility

Must be programmed if implemented.

MRS <Xt>, TRCSTALLCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSTALLCTLR;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSTALLCTLR;

```

MSR TRCSTALLCTLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSTALLCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSTALLCTLR = X[t, 64];

```

A.2.8.32 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

AArch64 register TRCIDR3 bits [31:0] are architecturally mapped to External register [B.2.2.7.36 TRCIDR3, ID Register 3](#) on page 2037 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x000	1101	x000	0111	xx00	0000	0000	0100
63	59	55	51	47	43	39	35	31	27	23	19	15	11	7	3



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-411: AARCH64_TRCIDR3 bit assignments

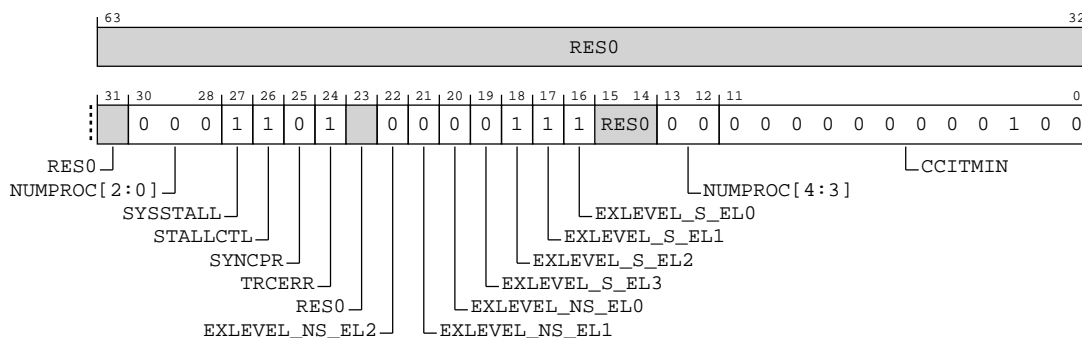


Table A-1052: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. 0b1 Stalling of the PE is permitted.	0b1
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. 0b1 Stalling of the PE is implemented.	0b1
[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 implemented. 0b0 Non-secure EL2 is not implemented.	0b0

Bits	Name	Description	Reset
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. 0b0 Non-secure EL1 is not implemented.	0b0
[20]	EXLEVEL_NS_ELO	Indicates if Non-secure ELO implemented. 0b0 Non-secure ELO is not implemented.	0b0
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. 0b0 Secure EL3 is not implemented.	0b0
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. 0b1 Secure EL2 is implemented.	0b1
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. 0b1 Secure EL1 is implemented.	0b1
[16]	EXLEVEL_S_ELO	Indicates if Secure ELO implemented. 0b1 Secure ELO is implemented.	0b1
[15:14]	RES0	Reserved	RES0
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. 0b000000 The trace unit can trace one PE. This field reads as 0b000000.	0b000000
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in AArch64-TRCCCCTLR.THRESHOLD. 0b0000000000100 Minimum allowed threshold value is 4.	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;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else

```

```
        X[t, 64] = TRCIDR3;
    elseif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = TRCIDR3;
```

A.2.8.33 TRCTSCTLR, Timestamp Control Register

Controls the insertion of global timestamps in the trace stream.

Configurations

AArch64 register TRCTSCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.8 TRCTSCTLR, Timestamp Control Register](#) on page 1989 bits [31:0].

Attributes

Width

64

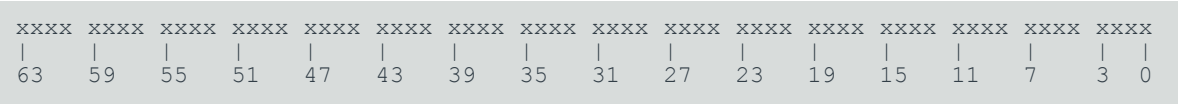
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-412: AARCH64_TRCTSCTLR bit assignments

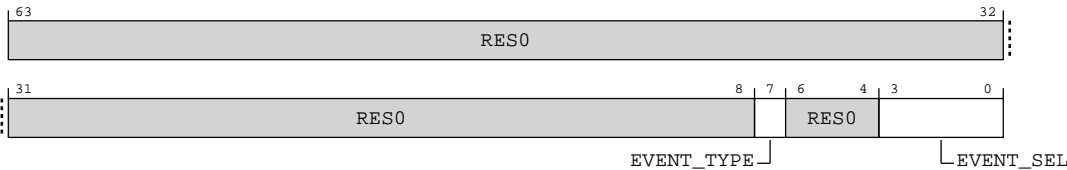


Table A-1054: TRCTSCTLR bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7]	EVENT_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. EVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. EVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. EVENT_TYPE controls whether EVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire.	xxxx

Access

Must be programmed if AArch64-TRCCONFIGR.TS == 0b1.

MRS <Xt>, TRCTSCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b000

MSR TRCTSCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1100	0b000

Accessibility

Must be programmed if AArch64-TRCCONFIGR.TS == 0b1.

MRS <Xt>, TRCTSCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCTSCTLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCTSCTLR;

```


MSR TRCTSCTLR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCTSCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCTSCTLR = X[t, 64];
```

A.2.8.34 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR4 bits [31:0] are architecturally mapped to External register [B.2.2.7.37 TRCIDR4, ID Register 4](#) on page 2039 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	0001	0010	0111	0000	xxx1	0010	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-413: AARCH64_TRCIDR4 bit assignments

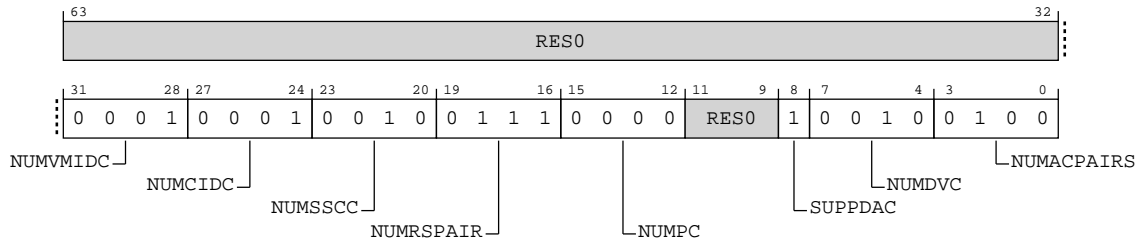


Table A-1057: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. 0b0001 The implementation has one Virtual Context Identifier Comparator.	0b0001
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. 0b0001 The implementation has one Context Identifier Comparator.	0b0001
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. 0b0010 The implementation has two Single-shot Comparator Controls.	0b0010
[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. 0b1 Data address comparisons implemented.	0b1
[7:4]	NUMDVC	Indicates the number of data value comparators. 0b0010 Two data value comparators implemented.	0b0010
[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;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR4;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR4;
```

A.2.8.35 TRCSYNCP, Synchronization Period Register

Controls how often trace protocol synchronization requests occur.

Configurations

AArch64 register TRCSYNCP bits [31:0] are architecturally mapped to External register [B.2.2.7.9 TRCSYNCP, Synchronization Period Register](#) on page 1990 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-414: AARCH64_TRCSYNCPR bit assignments

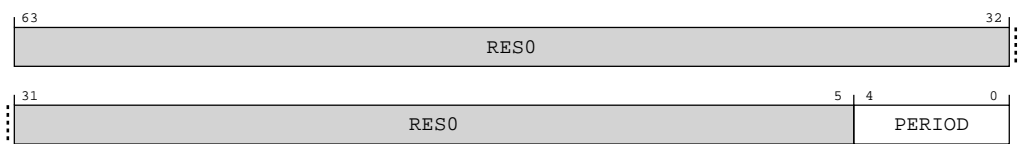


Table A-1059: TRCSYNCPR bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[4:0]	PERIOD	<p>Defines the number of bytes of trace between each periodic trace protocol synchronization request.</p> <p>0b000000 Trace protocol synchronization is disabled.</p> <p>0b010000 Trace protocol synchronization request occurs after 2^8 bytes of trace.</p> <p>0b01001 Trace protocol synchronization request occurs after 2^9 bytes of trace.</p> <p>0b01010 Trace protocol synchronization request occurs after 2^{10} bytes of trace.</p> <p>0b01011 Trace protocol synchronization request occurs after 2^{11} bytes of trace.</p> <p>0b01100 Trace protocol synchronization request occurs after 2^{12} bytes of trace.</p> <p>0b01101 Trace protocol synchronization request occurs after 2^{13} bytes of trace.</p> <p>0b01110 Trace protocol synchronization request occurs after 2^{14} bytes of trace.</p> <p>0b01111 Trace protocol synchronization request occurs after 2^{15} bytes of trace.</p> <p>0b10000 Trace protocol synchronization request occurs after 2^{16} bytes of trace.</p> <p>0b10001 Trace protocol synchronization request occurs after 2^{17} bytes of trace.</p> <p>0b10010 Trace protocol synchronization request occurs after 2^{18} bytes of trace.</p> <p>0b10011 Trace protocol synchronization request occurs after 2^{19} bytes of trace.</p> <p>0b10100 Trace protocol synchronization request occurs after 2^{20} bytes of trace.</p> <p>Other values are reserved. If a reserved value is programmed into PERIOD, then one of the following behaviors occurs:</p> <ul style="list-style-type: none"> If a reserved value less than 0b01000 is programmed, trace protocol synchronisation requests occur at approximately the specified period. If a reserved value greater than 0b10100 is programmed, no trace protocol synchronisation requests are generated. 	5 {x}

Access

Must be programmed if AArch64-TRCIDR3.SYNCPR == 0b0.

MRS <Xt>, TRCSYNCPR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b000

MSR TRCSYNCPR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b000

Accessibility

Must be programmed if AArch64-TRCIDR3.SYNCPR == 0b0.

MRS <Xt>, TRCSYNCPR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSYNCPR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSYNCPR;

```

MSR TRCSYNCPR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSYNCPR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSYNCPR = X[t, 64];

```

A.2.8.36 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

ConfigurationsAArch64 register TRCIDR5 bits [31:0] are architecturally mapped to External register [B.2.2.7.38 TRCIDR5, ID Register 5](#) on page 2041 bits [31:0].**Attributes****Width**

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0010	100x	1100	0111	xxxx	1000	0011	1010
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-415: AARCH64_TRCIDR5 bit assignments

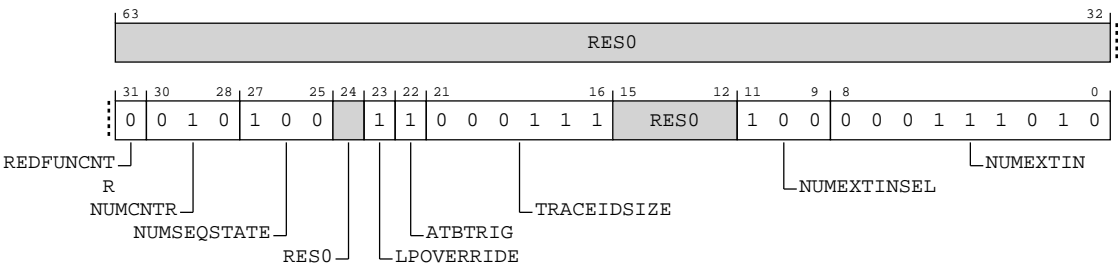


Table A-1062: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	REDFUNCNTR	Indicates if the reduced function counter is implemented. 0b0 The reduced function counter is not supported.	0b0
[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. 0b1 The trace unit support Low-power Override Mode.	0b1

Bits	Name	Description	Reset
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. 0b1 The implementation supports ATB triggers.	0b1
[21:16]	TRACEIDSIZE	Indicates the trace ID width. 0b000111 The implementation supports a 7-bit trace ID.	0b000111
[15:12]	RES0	Reserved	RES0
[11:9]	NUMEXTINSEL	Indicates how many external input selector resources are implemented. 0b100 4 external input selector resources are available.	0b100
[8:0]	NUMEXTIN	Indicates how many external inputs are implemented. 0b000111010 The implementation has 58 external inputs.	0b000111010

Access

MRS <Xt>, TRCIDR5

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1101	0b111

Accessibility

MRS <Xt>, TRCIDR5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR5;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR5;

```

A.2.8.37 TRCCCCTLR, Cycle Count Control Register

Set the threshold value for cycle counting.

Configurations

AArch64 register TRCCCCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.10 TRCCCCTLR, Cycle Count Control Register](#) on page 1993 bits [31:0].

Attributes

Width

64

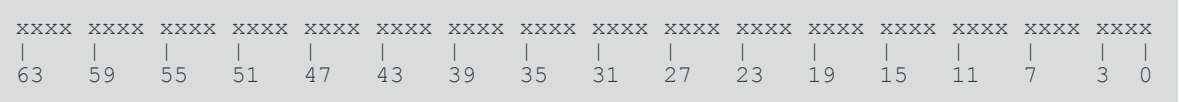
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-416: AARCH64_TRCCCCTLR bit assignments

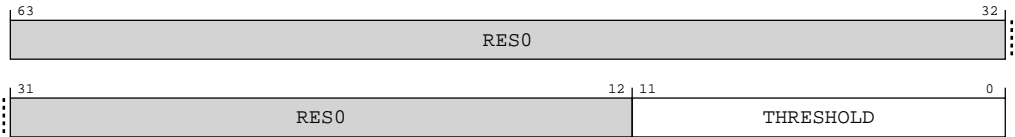


Table A-1064: TRCCCCTLR bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	RES0
[11:0]	THRESHOLD	<div>Sets the threshold value for instruction trace cycle counting.</div> <div>The minimum threshold value that can be programmed into THRESHOLD is given in AArch64-TRCIDR3.CCITMIN. If the THRESHOLD value is smaller than the value in AArch64-TRCIDR3.CCITMIN, then no cycle counts are generated in the trace and the cycle count threshold is not included in a traceinfo packet.</div> <div>Writing a value of zero when AArch64-TRCCONFIGR.CCI is set to enable instruction trace cycle counting, then no cycle counts are generated in the trace and the cycle count threshold is not included in a traceinfo packet.</div>	12 {x}

Access

Must be programmed if AArch64-TRCCONFIGR.CCI == 0b1.

MRS <Xt>, TRCCCCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b000

MSR TRCCCCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b000

Accessibility

Must be programmed if AArch64-TRCCONFIGR.CCI == 0b1.

MRS <Xt>, TRCCCCTLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCCCTLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCCCTLR;

```

MSR TRCCCCTLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCCCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCCCTLR = X[t, 64];

```

A.2.8.38 TRCIDR6, ID Register 6

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR6 bits [31:0] are architecturally mapped to External register [B.2.2.7.39 TRCIDR6, ID Register 6](#) on page 2042 bits [31:0].

Attributes

Width

64

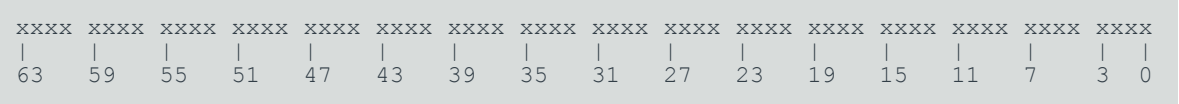
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-417: AARCH64_TRCIDR6 bit assignments

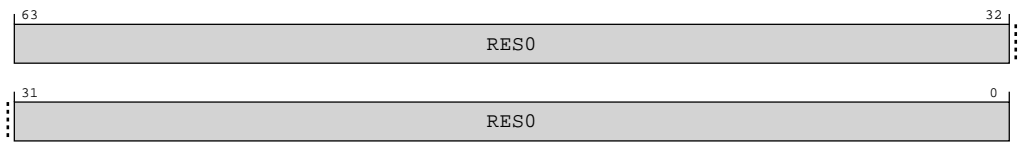


Table A-1067: TRCIDR6 bit descriptions

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

Access

MRS <Xt>, TRCIDR6

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1110	0b111

Accessibility

MRS <Xt>, TRCIDR6

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR6;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = TRCIDR6;
```

A.2.8.39 TRCBBCTLR, Branch Broadcast Control Register

Controls the regions in the memory map where branch broadcasting is active.

Configurations

AArch64 register TRCBBCTLR bits [31:0] are architecturally mapped to External register [B.2.2.7.11 TRCBBCTLR, Branch Broadcast Control Register](#) on page 1994 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	
63	59	55	51	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-418: AARCH64_TRCBBCTLR bit assignments

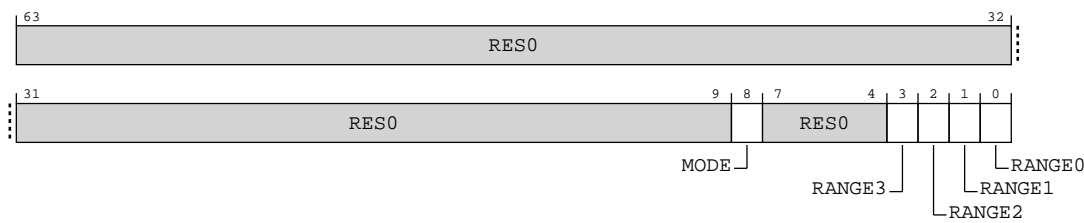


Table A-1069: TRCBBCTLR bit descriptions

Bits	Name	Description	Reset
[63:9]	RES0	Reserved	RES0
[8]	MODE	<p>Mode.</p> <p>0b0</p> <p>Exclude Mode.</p> <p>Branch broadcasting is not active for instructions in the address ranges defined by RANGE.</p> <p>If RANGE == 0x00 then branch broadcasting is active for all instructions.</p> <p>0b1</p> <p>Include Mode.</p> <p>Branch broadcasting is active for instructions in the address ranges defined by RANGE.</p> <p>If RANGE == 0x00 then no instructions are considered to be in the branch broadcasting region.</p>	x
[7:4]	RES0	Reserved	RES0
[3:0]	RANGE<m>, bit[m], where m = 3 to 0	<p>Address range field.</p> <p>Selects which Address Range Comparators are in use with branch broadcasting.</p> <p>0b0</p> <p>The address range that Address Range Comparator m defines, is not selected.</p> <p>0b1</p> <p>The address range that Address Range Comparator m defines, is selected.</p>	xxxx

Access

Must be programmed if AArch64-TRCCONFIGR.BB == 0b1.

MRS <Xt>, TRCBBCTLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b000

MSR TRCBBCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b000

Accessibility

Must be programmed if AArch64-TRCCONFIGR.BB == 0b1.

MRS <Xt>, TRCBBCTLR

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

```
if CPACR_EL1.TTA == '1' then
    AArch64.SystemAccessTrap(EL1, 0x18);
elseif CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    X[t, 64] = TRCBBCTLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCBBCTLR;
```

MSR TRCBBCTLR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCBBCTLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCBBCTLR = X[t, 64];
```

A.2.8.40 TRCIDR7, ID Register 7

Returns the tracing capabilities of the trace unit.

Configurations

AArch64 register TRCIDR7 bits [31:0] are architecturally mapped to External register [B.2.2.7.40 TRCIDR7, ID Register 7](#) on page 2044 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-419: AARCH64_TRCIDR7 bit assignments

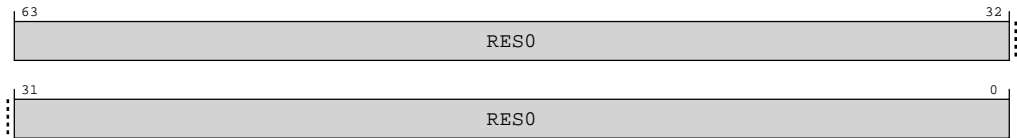


Table A-1072: TRCIDR7 bit descriptions

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

Access

MRS <Xt>, TRCIDR7

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b1111	0b111

Accessibility

MRS <Xt>, TRCIDR7

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR7;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCIDR7;
```

A.2.8.41 TRCSSCCR<n>, Single-shot Comparator Control Register <n> , n = 0 - 1

Controls the corresponding Single-shot Comparator Control resource.

Configurations

AArch64 register TRCSSCCR<n> bits [31:0] are architecturally mapped to External register [B.2.2.7.42 TRCSSCCR<n>, Single-shot Comparator Control Register <n> , n = 0 - 1](#) on page 2052 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	
63	59	55	51	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-420: AARCH64_TRCSSCCR<n> bit assignments

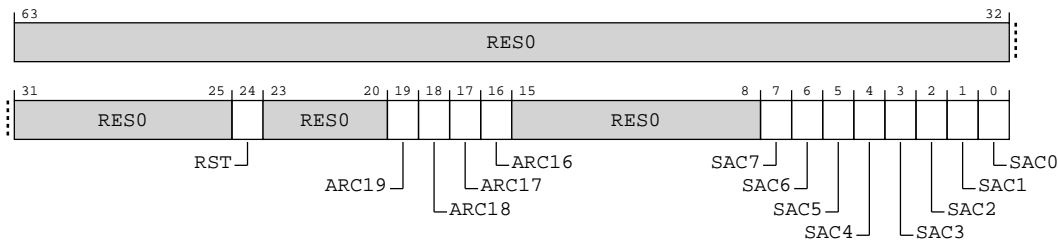


Table A-1074: TRCSSCCR<n> bit descriptions

Bits	Name	Description	Reset
[63:25]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[24]	RST	Selects the Single-shot Comparator Control mode. 0b0 The Single-shot Comparator Control is in single-shot mode. 0b1 The Single-shot Comparator Control is in multi-shot mode.	x
[23:20]	RES0	Reserved	RES0
[19:16]	ARC<m>, bit[m], where m = 19 to 16	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.	xxxx
[15:8]	RES0	Reserved	RES0
[7:0]	SAC<m>, bit[m], where m = 7 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.	8{x}

Access

Must be programmed if any TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 0b1.

MRS <Xt>, TRCSSCCR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	'0':m[2:0]	0b010

MSR TRCSSCCR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	'0':m[2:0]	0b010

Accessibility

Must be programmed if any TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 0b1.

MRS <Xt>, TRCSSCCR<m>

```
integer m = UInt(CRm<2:0>);
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```
        X[t, 64] = TRCSSCCR[m];
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = TRCSSCCR[m];
```

MSR TRCSSCCR<m>, <Xt>

```
integer m = UInt(CRm<2:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSSCCR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSSCCR[m] = X[t, 64];
```

A.2.8.42 TRCSSCSR<n>, Single-shot Comparator Control Status Register <n> , n = 0 - 1

Returns the status of the corresponding Single-shot Comparator Control.

Configurations

AArch64 register TRCSSCSR<n> bits [31:0] are architecturally mapped to External register [B.2.2.7.43 TRCSSCSR<n>, Single-shot Comparator Control Status Register <n> , n = 0 - 1](#) on page 2053 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

RO

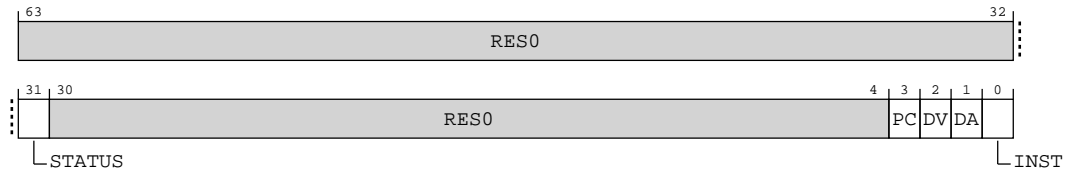
Reset value

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

**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-421: AARCH64_TRCSSCSR<n> bit assignments**Table A-1077: TRCSSCSR<n> bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	STATUS	<p>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 and AArch64-TRCSSCCR<n>.SAC.</p> <p>0b0</p> <p>No match has occurred. When the first match occurs, this field takes a value of 0b1. It remains at 0b1 until explicitly modified by a write to this register.</p> <p>0b1</p> <p>One or more matches has occurred. If AArch64-TRCSSCCR<n>.RST == 0b0 then:</p> <ul style="list-style-type: none"> There is only one match and no more matches are possible. Software must reset this bit to 0b0 to re-enable the Single-shot Comparator Control. <p>The reset value is UNKNOWN. STATUS must be written to set an initial state when programming the trace unit, if the single-shot comparator is to be used.</p>	x
[30:4]	RES0	Reserved	RES0
[3]	PC	<p>PE Comparator Input support.</p> <p>0b0</p> <p>This Single-shot Comparator Control does not support PE Comparator Inputs.</p>	x
[2]	DV	<p>Data value comparator support.</p> <p>0b0</p> <p>This Single-shot Comparator Control does not support data value comparisons. TRCSSCSR0 has this value.</p> <p>0b1</p> <p>This Single-shot Comparator Control supports data value comparisons. TRCSSCSR1 has this value.</p>	x

Bits	Name	Description	Reset
[1]	DA	Data address comparator support. 0b0 This Single-shot Comparator Control does not support data address comparisons. TRCSSCSR0 has this value. 0b1 This Single-shot Comparator Control supports data address comparisons. TRCSSCSR1 has this value.	x
[0]	INST	Instruction address comparator support. 0b0 This Single-shot Comparator Control does not support instruction address comparisons. TRCSSCSR1 has this value. 0b1 This Single-shot Comparator Control supports instruction address comparisons. TRCSSCSR0 has this value.	x

Access

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 0b1.

MRS <Xt>, TRCSSCSR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	'1':m[2:0]	0b010

MSR TRCSSCSR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	'1':m[2:0]	0b010

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 0b1.

MRS <Xt>, TRCSSCSR<m>

```

integer m = UInt(CRm<2:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSSCSR[m];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCSSCSR[m];

```

MSR TRCSSCSR<m>, <Xt>

```
integer m = UInt(CRm<2:0>);
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSSCSR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCSSCSR[m] = X[t, 64];
```

A.2.8.43 TRCOSLAR, Trace OS Lock Access Register

Controls whether the Trace OS Lock is locked.

Configurations

AArch64 register TRCOSLAR bits [31:0] are architecturally mapped to External register [B.2.2.7.44 TRCOSLAR, Trace OS Lock Access Register](#) on page 2056 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-422: AARCH64_TRCOSLAR bit assignments

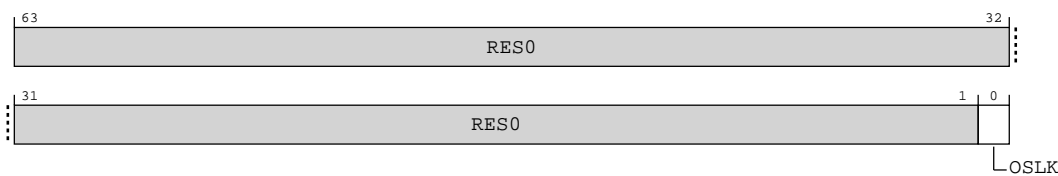


Table A-1080: TRCOSLAR bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	RES0
[0]	OSLK	OS Lock control bit. 0b0 Unlocks the OS Lock. 0b1 Locks the OS Lock. This setting disables the trace unit.	x

Access

MSR TRCOSLAR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0000	0b100

Accessibility

MSR TRCOSLAR, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCOSLAR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCOSLAR = X[t, 64];
```

A.2.8.44 TRCOSLSR, Trace OS Lock Status Register

Returns the status of the Trace OS Lock.

Configurations

AArch64 register TRCOSLSR bits [31:0] are architecturally mapped to External register [B.2.2.7.45 TRCOSLSR, Trace OS Lock Status Register](#) on page 2057 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1x10
63	59	55	51	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-423: AARCH64_TRCOSLSR bit assignments

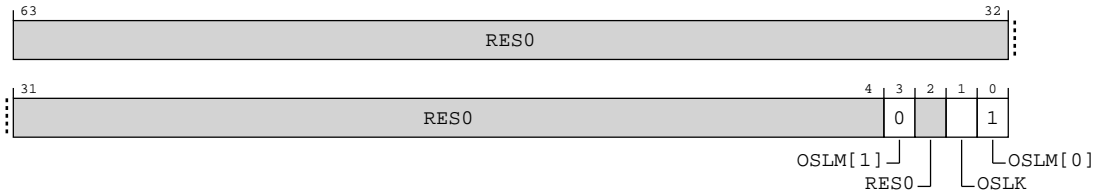


Table A-1082: TRCOSLSR bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3, 0]	OSLM	OS Lock model. 0b10 Trace OS Lock is implemented. Access to this field is: RO	0b10

Bits	Name	Description	Reset
[2]	RES0	Reserved	RES0
[1]	OSLK	OS Lock status. 0b0 The OS Lock is unlocked. 0b1 The OS Lock is locked.	0b1

Access

MRS <Xt>, TRCOSLSR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	0b0001	0b100

Accessibility

MRS <Xt>, TRCOSLSR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCOSLSR;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCOSLSR;

```

A.2.8.45 TRCRSCTLR<n>, Resource Selection Control Register <n> , n = 2 - 15

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.

AArch64 register TRCRSCTLR<n> bits [31:0] are architecturally mapped to External register [B.2.2.7.41 TRCRSCTLR<n>, Resource Selection Control Register <n> , n = 2 - 15](#) on page 2045 bits [31:0].

Attributes

Width

64

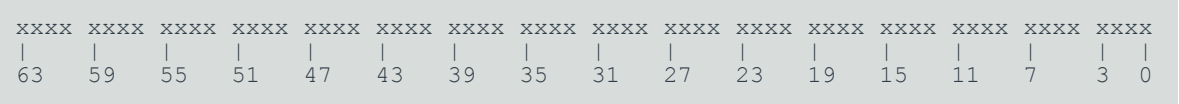
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-424: AARCH64_TRCRSCTLR<n> bit assignments

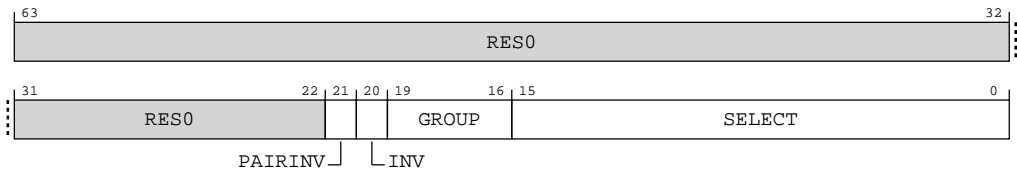


Table A-1084: TRCRSCTLR<n> bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[21]	PAIRINV	For TRCRSCTLR<n>, where n is even, 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	<p>Controls whether the resource, that GROUP and SELECT selects, is inverted.</p> <p>0b0 Do not invert the output of this selector.</p> <p>0b1 Invert the output of this selector.</p> <p>If:</p> <ul style="list-style-type: none"> A is the register TRCRSCTLR<m> where m is even. B is the register TRCRSCTLR<m+1>. <p>Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows:</p> <ul style="list-style-type: none"> 0b000 -> A and B. 0b001 -> RESERVED. 0b010 -> not(A) and B. 0b011 -> not(A) and not(B). 0b100 -> not(A) or not(B). 0b101 -> not(A) or B. 0b110 -> RESERVED. 0b111 -> A or B. 	x
[19:16]	GROUP	<p>Selects a group of resources.</p> <p>0b0000 External input selectors.</p> <p>0b0001 PE Comparator Inputs.</p> <p>0b0010 Counters and Sequencer.</p> <p>0b0011 Single-shot Comparator Controls.</p> <p>0b0100 Single Address Comparators.</p> <p>0b0101 Address Range Comparators.</p> <p>0b0110 Context Identifier Comparators.</p> <p>0b0111 Virtual Context Identifier Comparators.</p> <p>All other values are reserved.</p>	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.	16 {x}
[15:0 continued]	SELECT	SELECT encoding for PE Comparator Inputs 15:0 Reserved, RES0.	16 {x}
[15:0 continued]	SELECT	SELECT encoding for Counters and Sequencer 15:8 Reserved, RES0. SEQUENCER<m>, bit[m], for m = 7 to 4 Sequencer states. 0b0 Ignore Sequencer state m. 0b1 Select Sequencer state m. 3:2 Reserved, RES0.	16 {x}
[15:0 continued]	SELECT	COUNTERS<m>, bit[m], for m = 1 to 0 Counters resources at zero. 0b0 Ignore Counter m. 0b1 Select Counter m is zero.	16 {x}
[15:0 continued]	SELECT	SELECT encoding for Single-shot Comparator Controls 15:2 Reserved, RES0. SINGLE_SHOT<m>, bit[m], for m = 1 to 0 Selects one or more Single-shot Comparator Controls. 0b0 Ignore Single-shot Comparator Control m. 0b1 Select Single-shot Comparator Control m.	16 {x}

Bits	Name	Description	Reset
[15:0 continued]	SELECT	SELECT encoding for Single Address Comparators 15:8 Reserved, RES0. SAC<m>, bit[m], for m = 7 to 0 Selects one or more Single Address Comparators. 0b0 Ignore Single Address Comparator m. 0b1 Select Single Address Comparator m.	16 {x}
[15:0 continued]	SELECT	SELECT encoding for Address Range Comparators 15:4 Reserved, RES0. ARC<m>, bit[m], for m = 3 to 0 Selects one or more Address Range Comparators. 0b0 Ignore Address Range Comparator m. 0b1 Select Address Range Comparator m.	16 {x}
[15:0 continued]	SELECT	SELECT encoding for Context Identifier Comparators 15:1 Reserved, RES0. CID<m>, bit[m], for m = 0 Selects one or more Context Identifier Comparators. 0b0 Ignore Context Identifier Comparator m. 0b1 Select Context Identifier Comparator m.	16 {x}
[15:0 continued]	SELECT	SELECT encoding for Virtual Context Identifier Comparators 15:1 Reserved, RES0. VMID<m>, bit[m], for m = 0 Selects one or more Virtual Context Identifier Comparators. 0b0 Ignore Virtual Context Identifier Comparator m. 0b1 Select Virtual Context Identifier Comparator m.	16 {x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT0.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT0.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT0.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT1.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT1.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT1.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT2.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT2.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT2.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT3.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT3.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT3.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0b0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 0b1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0b0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 0b1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEV.RST.TYPE == 0b0 and AArch64-TRCSEQRSTEV.RST.SEL == n.
- AArch64-TRCSEQRSTEV.RST.TYPE == 0b1 and AArch64-TRCSEQRSTEV.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0b0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0b1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0b0 and AArch64-TRCVICTLR.EVENT.SEL == n.
- AArch64-TRCVICTLR.EVENT.TYPE == 0b1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.

MRS <Xt>, TRCRSCTLR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	m[3:0]	'00':m[4]

MSR TRCRSCTLR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0001	m[3:0]	'00':m[4]

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT0.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT0.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT0.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT0.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT1.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT1.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT1.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT1.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT2.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT2.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT2.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT2.SEL == n/2.
- AArch64-TRCEVENTCTLOR.EVENT3.TYPE == 0b0 and AArch64-TRCEVENTCTLOR.EVENT3.SEL == n.
- AArch64-TRCEVENTCTLOR.EVENT3.TYPE == 0b1 and AArch64-TRCEVENTCTLOR.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0b0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 0b1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0b0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 0b1 and TRCSEQEVR<a>.F.SEL = n/2.
- AArch64-TRCSEQRSTEV.RST.TYPE == 0b0 and AArch64-TRCSEQRSTEV.RST.SEL == n.
- AArch64-TRCSEQRSTEV.RST.TYPE == 0b1 and AArch64-TRCSEQRSTEV.RST.SEL == n/2.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0b0 and AArch64-TRCTSCTLR.EVENT.SEL == n.
- AArch64-TRCTSCTLR.EVENT.TYPE == 0b1 and AArch64-TRCTSCTLR.EVENT.SEL == n/2.
- AArch64-TRCVICTLR.EVENT.TYPE == 0b0 and AArch64-TRCVICTLR.EVENT.SEL == n.

- AArch64-TRCVICTLR.EVENT.TYPE == 0b1 and AArch64-TRCVICTLR.EVENT.SEL == n/2.
MRS <Xt>, TRCRSCTLR<m>

```
integer m = UInt(op2<0>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCRSCTLR[m];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCRSCTLR[m];
```

MSR TRCRSCTLR<m>, <Xt>

```
integer m = UInt(op2<0>:CRm<3:0>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCRSCTLR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCRSCTLR[m] = X[t, 64];
```

A.2.8.46 TRCDVCVR<n>, Data Value Comparator Value Register <n> , n = 0 - 1

Contains a data value.

Configurations

AArch64 register TRCDVCVR<n> bits [63:0] are architecturally mapped to External register [B.2.2.7.50 TRCDVCVR<n>, Data Value Comparator Value Register <n> , n = 0 - 1](#) on page 2067 bits [63:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-425: AARCH64_TRCDVCVR<n> bit assignments

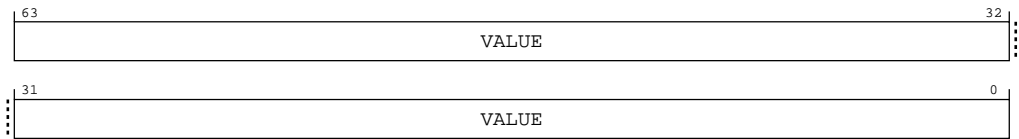


Table A-1087: TRCDVCVR<n> bit descriptions

Bits	Name	Description	Reset
[63:0]	VALUE	Data Value. The data value comparators can support implementations that use multiple data widths. When the trace unit compares the VALUE field with a data value that has a width less than this field, then software must also write the comparison data value to both the upper bits and the lower bits of the VALUE field. For example, in a system that supports both 32-bit and 64-bit data widths, then software must set VALUE[63:32]==VALUE[31:0] if the trace unit is to compare a 32-bit data value.	64 {x}

Access

Might ignore writes when the trace unit is enabled or not idle.

MRS <Xt>, TRCDVCVR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[1:0]:'00'	'10':m[2]

MSR TRCDVCVR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[1:0]:'00'	'10':m[2]

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

MRS <Xt>, TRCDVCVR<m>

```

integer m = UInt(op2<0>:CRm<3:2>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCDVCVR[m];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCDVCVR[m];

```

MSR TRCDVCVR<m>, <Xt>

```

integer m = UInt(op2<0>:CRm<3:2>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCDVCVR[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCDVCVR[m] = X[t, 64];

```

A.2.8.47 TRCDVCMR<n>, Data Value Comparator Mask Register <n> , n = 0 - 1

Contains a data mask value.

Configurations

AArch64 register TRCDVCMR<n> bits [63:0] are architecturally mapped to External register [B.2.2.7.51 TRCDVCMR<n>, Data Value Comparator Mask Register <n> , n = 0 - 1](#) on page 2069 bits [63:0].

Attributes

Width

64

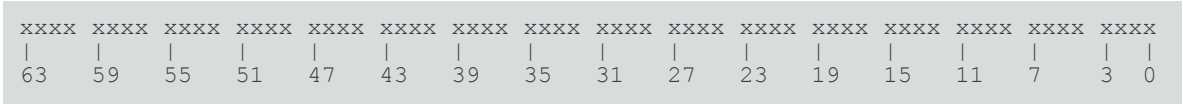
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-426: AArch64_TRCDVCMR<n> bit assignments

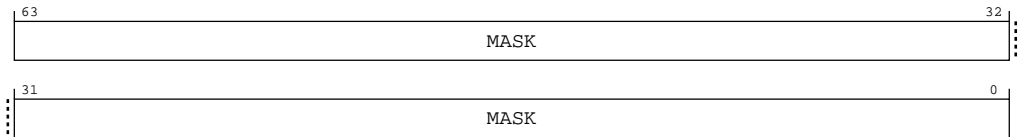


Table A-1090: TRCDVCMR<n> bit descriptions

Bits	Name	Description	Reset
[63:0]	MASK	<p>Data mask value.</p> <p>If a bit is set to 1 in the mask then the comparator ignores that bit number for a data value comparison. Software must ensure that the relevant bit in AArch64-TRCDVCVR<n> is programmed to 0, otherwise the comparator might fail to match.</p> <p>The data value comparators can support implementations that use multiple data widths. When the trace unit compares the AArch64-TRCDVCVR<n>.VALUE field with a data value that has a width less than this field, then software must also write the data mask value to both the upper bits and the lower bits of the MASK field. For example, in a system that supports both 32-bit and 64-bit data widths, then software must set MASK[63:32]=MASK[31:0] if the trace unit is to compare a 32-bit data value.</p>	64 {x}

Access

Might ignore writes when the trace unit is enabled or not idle.

MRS <Xt>, TRCDVCMR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[1:0]:'00'	'11':m[2]

MSR TRCDVCMR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[1:0]:'00'	'11':m[2]

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

MRS <Xt>, TRCDVCMR<m>

```

integer m = UInt(op2<0>:CRm<3:2>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCDVCMR[m];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCDVCMR[m];

```

MSR TRCDVCMR<m>, <Xt>

```

integer m = UInt(op2<0>:CRm<3:2>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCDVCMR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCDVCMR[m] = X[t, 64];

```

A.2.8.48 TRCACVR<n>, Address Comparator Value Register <n> , n = 0 - 7

Contains the address value.

Configurations

AArch64 register TRCACVR<n> bits [63:0] are architecturally mapped to External register [B.2.2.7.48 TRCACVR<n>, Address Comparator Value Register <n> , n = 0 - 7](#) on page 2061 bits [63:0].

Attributes

Width

64

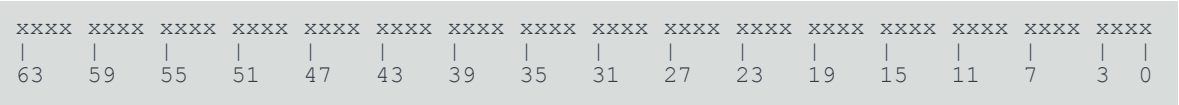
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-427: AARCH64_TRCACVR<n> bit assignments

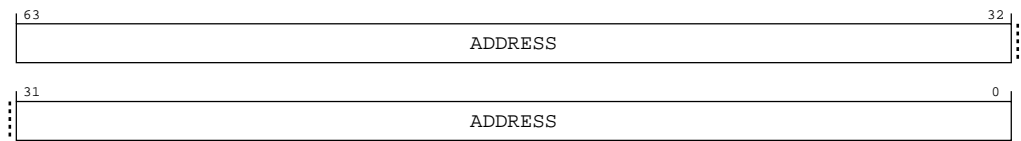


Table A-1093: TRCACVR<n> bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	<p>Address Value.</p> <p>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.</p> <p>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 virtual address size supported by the PE.</p> <p>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.</p>	64 {x}

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 0b1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 0b1.

- AArch64-TRCVISSCTLR.START[n] == 0b1.
- AArch64-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.
- TRCSSCCR<>.SAC[n] == 0b1.
- AArch64-TRCQCTLR.RANGE[n/2] == 0b1.

MRS <Xt>, TRCACVR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:'0'	'00':m[3]

MSR TRCACVR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:'0'	'00':m[3]

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- AArch64-TRCVIICTLR.EXCLUDE[n/2] == 0b1.
- AArch64-TRCVIICTLR.INCLUDE[n/2] == 0b1.
- AArch64-TRCVISSCTLR.START[n] == 0b1.
- AArch64-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.
- TRCSSCCR<>.SAC[n] == 0b1.
- AArch64-TRCQCTLR.RANGE[n/2] == 0b1.

MRS <Xt>, TRCACVR<m>

```
integer m = UInt(op2<0>:CRm<3:1>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCACVR[m];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```

else
    X[t, 64] = TRCACVR[m];

```

MSR TRCACVR<m>, <Xt>

```

integer m = UInt(op2<0>:CRm<3:1>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCACVR[m] = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCACVR[m] = X[t, 64];

```

A.2.8.49 TRCACATR<n>, Address Comparator Access Type Register <n> , n = 0 - 7

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

AArch64 register TRCACATR<n> bits [63:0] are architecturally mapped to External register [B.2.2.7.49 TRCACATR<n>, Address Comparator Access Type Register <n> , n = 0 - 7](#) on page 2063 bits [63:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

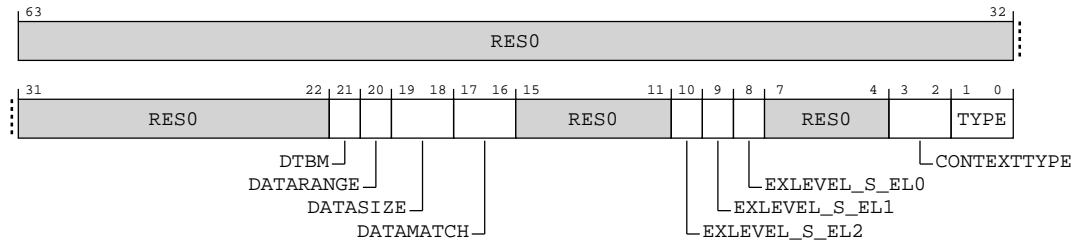
Reset value

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

**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-428: AARCH64_TRCACATR<n> bit assignments**Table A-1096: TRCACATR<n> bit descriptions**

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[21]	DTBM	Controls whether data address comparisons use the data address [63:56] bits. 0b0 The trace unit ignores the data address [63:56] bits for data address comparisons. 0b1 The trace unit uses the data address [63:56] bits for data address comparisons.	x
[20]	DATARANGE	Controls whether a data value comparison uses the single address comparator or the address range comparator. 0b0 The trace unit uses the single address comparator for data value comparisons. The address range comparator may match at any time. The other single address comparator in the pair is not affected. 0b1 The trace unit uses the address range comparator for data value comparisons. The single address comparators in this pair may match at any time. The trace unit ignores this field when DATAMATCH==0b00.	x
[19:18]	DATASIZE	Controls the width of the data value comparison. 0b00 Byte. 0b01 Halfword. 0b10 Word. 0b11 Doubleword.	xx

Bits	Name	Description	Reset
[17:16]	DATAMATCH	Controls how the trace unit performs a data value comparison. 0b00 The trace unit does not perform a data value comparison. 0b01 The trace unit performs a data value comparison. If the data value comparator matches and the address comparator matches, the trace unit signals a match. 0b10 Reserved. 0b11 The trace unit performs a data value comparison. If the data value comparator does not match and the address comparator matches, the trace unit signals a match.	xx
[15:11]	RES0	Reserved	RES0
[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_ELO	Secure ELO address comparison control. Controls whether a comparison can occur at ELO in Secure state. 0b0 The address comparator performs comparisons in Secure ELO. 0b1 The address comparator does not perform comparisons in Secure ELO.	x
[7:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:2]	CONTEXTTYPE	<p>Controls whether the address comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparisons.</p> <p>0b00</p> <p>The address comparator is not dependent on the Context Identifier Comparator or Virtual Context Identifier Comparator.</p> <p>0b01</p> <p>The address comparator is dependent on the Context Identifier Comparator. If both the Context Identifier Comparator and the address comparison match, the address comparator signals a match.</p> <p>0b10</p> <p>The address comparator is dependent on the Virtual Context Identifier Comparator. If both the Virtual Context Identifier Comparator and the address comparison match, the address comparator signals a match.</p> <p>0b11</p> <p>The address comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator. If the Context Identifier Comparator, the Virtual Context Identifier Comparator and address comparison all match, the address comparator signals a match.</p>	xx
[1:0]	TYPE	<p>Controls what type of comparison the trace unit performs.</p> <p>0b00</p> <p>Instruction address.</p> <p>0b01</p> <p>Data load address.</p> <p>0b10</p> <p>Data store address.</p> <p>0b11</p> <p>Data load address or data store address.</p>	xx

Access

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 0b1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 0b1.
- AArch64-TRCVISSCTLR.START[n] == 0b1.
- AArch64-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.
- TRCSSCCR<>.SAC[n] == 0b1.

- AArch64-TRCQCTLR.RANGE[n/2] == 0b1.

MRS <Xt>, TRCACATR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:'0'	'01':m[3]

MSR TRCACATR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0010	m[2:0]:'0'	'01':m[3]

Accessibility

Must be programmed if any of the following are true:

- AArch64-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- AArch64-TRCVIIECTLR.EXCLUDE[n/2] == 0b1.
- AArch64-TRCVIIECTLR.INCLUDE[n/2] == 0b1.
- AArch64-TRCVISSCTLR.START[n] == 0b1.
- AArch64-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.
- TRCSSCCR<>.SAC[n] == 0b1.
- AArch64-TRCQCTLR.RANGE[n/2] == 0b1.

MRS <Xt>, TRCACATR<m>

```
integer m = UInt(op2<0>:CRm<3:1>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCACATR[m];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCACATR[m];
```

MSR TRCACATR<m>, <Xt>

```
integer m = UInt(op2<0>:CRm<3:1>);

if PSTATE.EL == EL0 then
```

```
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
      AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      TRCACATR[m] = X[t, 64];
  elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      TRCACATR[m] = X[t, 64];
```

A.2.8.50 TRCCIDCCTLRO, Context Identifier Comparator Control Register 0

Contains Context identifier mask values for the AArch64-TRCCIDCVR<n> registers, for n = 0 to 3.

Configurations

AArch64 register TRCCIDCCTLRO bits [31:0] are architecturally mapped to External register [B.2.2.7.54 TRCCIDCCTLRO, Context Identifier Comparator Control Register 0](#) on page 2074 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-429: AARCH64_TRCCIDCCTLR0 bit assignments

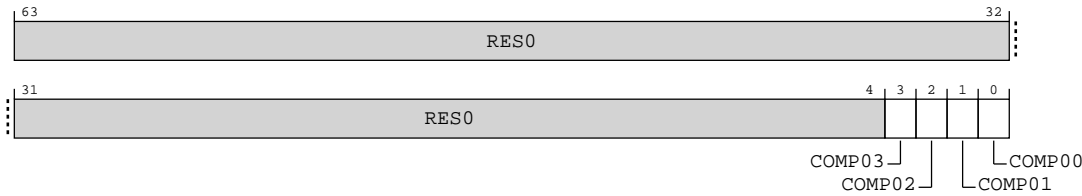


Table A-1099: TRCCIDCCTLR0 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:0]	COMP0<m>, bit[m], where m = 3 to 0	<p>TRCCIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0.</p> <p>0b0</p> <p>The trace unit includes TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.</p> <p>0b1</p> <p>The trace unit ignores TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.</p>	xxxx

Access

If software uses the AArch64-TRCCIDCVR<n> registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in AArch64-TRCCIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in AArch64-TRCCIDCVR<n> is not 0x00, the Context Identifier Comparator will not match.

MRS <Xt>, TRCCIDCCTLR0

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b010

MSR TRCCIDCCTLR0, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0000	0b010

Accessibility

If software uses the AArch64-TRCCIDCVR<n> registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in AArch64-TRCCIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in AArch64-TRCCIDCVR<n> is not 0x00, the Context Identifier Comparator will not match.

MRS <Xt>, TRCCIDCCTLRO

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCIDCCTLRO;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCIDCCTLRO;
```

MSR TRCCIDCCTLRO, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCIDCCTLRO = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCIDCCTLRO = X[t, 64];
```

A.2.8.51 TRCVMIDCCTLRO, Virtual Context Identifier Comparator Control Register 0

Virtual Context Identifier Comparator mask values for the AArch64-TRCVMIDCVR<n> registers, where n=0-3.

Configurations

AArch64 register TRCVMIDCCTLRO bits [31:0] are architecturally mapped to External register [B.2.2.7.55 TRCVMIDCCTLRO, Virtual Context Identifier Comparator Control Register 0](#) on page 2075 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
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-430: AARCH64_TRCVMIDCCTLR0 bit assignments

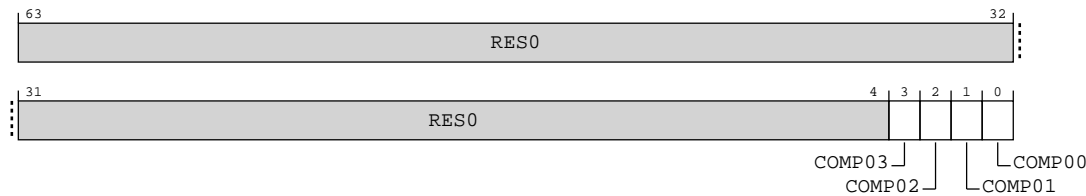


Table A-1102: TRCVMIDCCTLR0 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:0]	COMP0<m>, bit[m], where m = 3 to 0	<p>TRCVMIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0.</p> <p>0b0</p> <p>The trace unit includes TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.</p> <p>0b1</p> <p>The trace unit ignores TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.</p>	xxxx

Access

If software uses the AArch64-TRCVMIDCVR<n> registers, where n=0-3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in AArch64-TRCVMIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in AArch64-TRCVMIDCVR<n> is not 0x00, the Virtual Context Identifier Comparator will not match.

MRS <Xt>, TRCVMIDCCTLRO

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0010	0b010

MSR TRCVMIDCCTLRO, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	0b0010	0b010

Accessibility

If software uses the AArch64-TRCVMIDCVR<n> registers, where n=0-3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in AArch64-TRCVMIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in AArch64-TRCVMIDCVR<n> is not 0x00, the Virtual Context Identifier Comparator will not match.

MRS <Xt>, TRCVMIDCCTLRO

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVMIDCCTLRO;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVMIDCCTLRO;

```

MSR TRCVMIDCCTLRO, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVMIDCCTLRO = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVMIDCCTLRO = X[t, 64];

```

A.2.8.52 TRCCIDCVR<n>, Context Identifier Comparator Value Registers <n> , n = 0

Contains a Context identifier value.

Configurations

AArch64 register TRCCIDCVR<n> bits [63:0] are architecturally mapped to External register [B.2.2.7.52 TRCCIDCVR<n>, Context Identifier Comparator Value Registers <n> , n = 0](#) on page 2070 bits [63:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-431: AARCH64_TRCCIDCVR<n> bit assignments

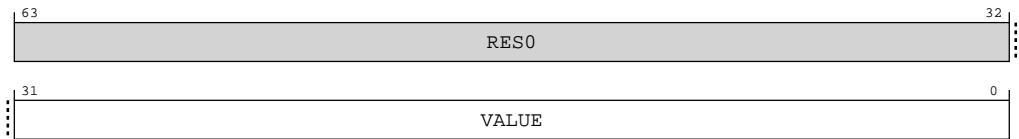


Table A-1105: TRCCIDCVR<n> bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	VALUE	Context identifier value. After a PE Reset, the trace unit assumes that the Context identifier is zero until the PE updates the Context identifier.	32 {x}

Access

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0110 and TRCRSCTLR<a>.CID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b01 or 0b11 and TRCACATR<a>.CONTEXT == n.

MRS <Xt>, TRCCIDCVR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:'0'	0b000

MSR TRCCIDCVR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:'0'	0b000

Accessibility

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0110 and TRCRSCTLR<a>.CID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b01 or 0b11 and TRCACATR<a>.CONTEXT == n.

MRS <Xt>, TRCCIDCVR<m>

```

integer m = UInt(CRm<3:1>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCIDCVR[m];
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = TRCCIDCVR[m];

```

MSR TRCCIDCVR<m>, <Xt>

```

integer m = UInt(CRm<3:1>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCIDCVR[m] = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then

```

```
AArch64.SystemAccessTrap(EL2, 0x18);  
else  
    TRCCIDCVR[m] = X[t, 64];
```

A.2.8.53 TRCVMIDCVR<n>, Virtual Context Identifier Comparator Value Register <n> , n = 0

Contains the Virtual Context Identifier Comparator value.

Configurations

AArch64 register TRCVMIDCVR<n> bits [63:0] are architecturally mapped to External register [B.2.2.7.53 TRCVMIDCVR<n>, Virtual Context Identifier Comparator Value Register <n> , n = 0](#) on page 2072 bits [63:0].

Attributes

Width

64

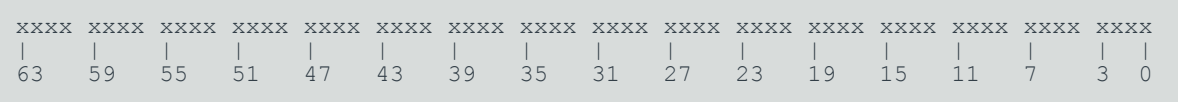
Functional group

Trace unit registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-432: AARCH64_TRCVMIDCVR<n> bit assignments

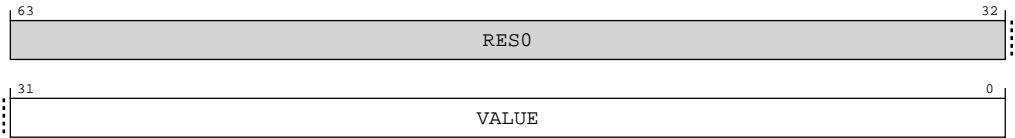


Table A-1108: TRCVMIDCVR<n> bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[31:0]	VALUE	Virtual context identifier value. After a PE Reset, the trace unit assumes that the Virtual context identifier is zero until the PE updates the Virtual context identifier.	32 {x}

Access

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0111 and TRCRSCTLR<a>.VMID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b10 or 0b11 and TRCACATR<a>.CONTEXT == n.

MRS <Xt>, TRCVMICDVR<m>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:'0'	0b001

MSR TRCVMICDVR<m>, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0011	m[2:0]:'0'	0b001

Accessibility

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0111 and TRCRSCTLR<a>.VMID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b10 or 0b11 and TRCACATR<a>.CONTEXT == n.

MRS <Xt>, TRCVMICDVR<m>

```
integer m = UInt(CRm<3:1>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCVMICDVR[m];
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = TRCVMICDVR[m];
```

MSR TRCVMICDVR<m>, <Xt>

```
integer m = UInt(CRm<3:1>);

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
```

```
elseif CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    TRCVMIDCVR[m] = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCVMIDCVR[m] = X[t, 64];
```

A.2.8.54 TRCDEVID, Device Configuration Register

Provides discovery information for the component.

Configurations

AArch64 register TRCDEVID bits [31:0] are architecturally mapped to External register [B.2.2.7.64 TRCDEVID, Device Configuration Register](#) on page 2088 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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-433: AARCH64_TRCDEVID bit assignments

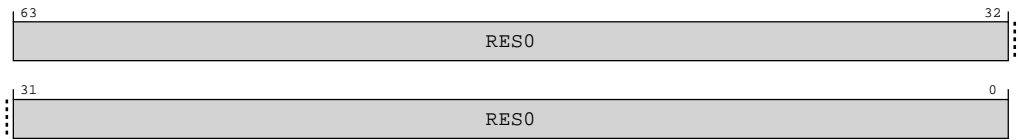


Table A-1111: TRCDEVID bit descriptions

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

Access

MRS <Xt>, TRCDEVID

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b0010	0b111

Accessibility

MRS <Xt>, TRCDEVID

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCDEVID;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCDEVID;

```

A.2.8.55 TRCCLAIMSET, Claim Tag Set Register

In conjunction with AArch64-TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

Configurations

AArch64 register TRCCLAIMSET bits [31:0] are architecturally mapped to External register [B.2.2.7.57 TRCCLAIMSET, Claim Tag Set Register](#) on page 2078 bits [31:0].

Attributes**Width**

64

Functional group

Trace unit registers

Access type

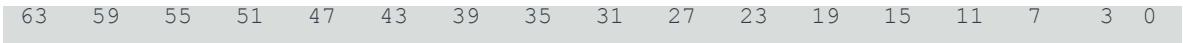
RAOW1S

Reset value

```

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 1111
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-434: AARCH64_TRCCLAIMSET bit assignments

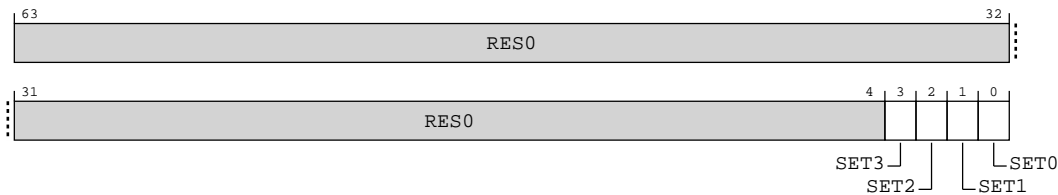


Table A-1113: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:0]	SET<m>, bit[m], where m = 3 to 0	<div>Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1.</div> <div>0b0 On a read: Claim Tag bit m is not implemented. On a write: Ignored.</div> <div>0b1 On a read: Claim Tag bit m is implemented. On a write: Set Claim Tag bit m to 0b1.</div>	0b1111

Access

MRS <Xt>, TRCCLAIMSET

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1000	0b110

MSR TRCCLAIMSET, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1000	0b110

Accessibility

MRS <Xt>, TRCCLAIMSET

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCLAIMSET;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCLAIMSET;

```

MSR TRCCLAIMSET, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCLAIMSET = X[t, 64];
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCLAIMSET = X[t, 64];

```

A.2.8.56 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with AArch64-TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

Configurations

AArch64 register TRCCLAIMCLR bits [31:0] are architecturally mapped to External register [B.2.2.7.58 TRCCLAIMCLR, Claim Tag Clear Register](#) on page 2080 bits [31:0].

Attributes

Width

64

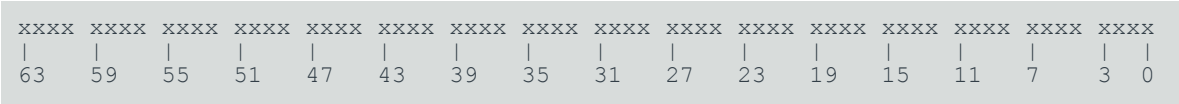
Functional group

Trace unit registers

Access type

RW1C

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-435: AARCH64_TRCCLAIMCLR bit assignments

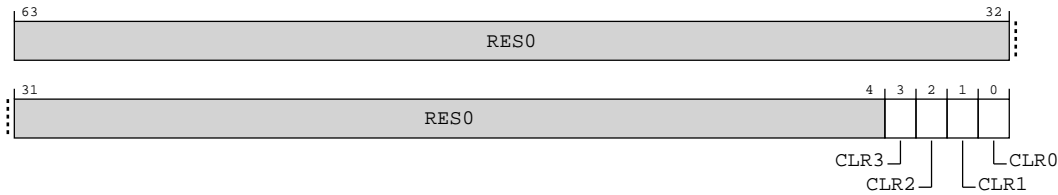


Table A-1116: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	RES0
[3:0]	CLR<m>, bit[m], where m = 3 to 0	<p>Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0.</p> <p>0b0</p> <p>On a read: Claim Tag bit m is not set.</p> <p>On a write: Ignored.</p> <p>0b1</p> <p>On a read: Claim Tag bit m is set.</p> <p>On a write: Clear Claim tag bit m to 0b0.</p> <p>The number of Claim Tag bits implemented is indicated in AArch64-TRCCLAIMSET.</p>	xxxx

Access

MRS <Xt>, TRCCLAIMCLR

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1001	0b110

MSR TRCCLAIMCLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1001	0b110

Accessibility

MRS <Xt>, TRCCLAIMCLR

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCLAIMCLR;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCCLAIMCLR;

```

MSR TRCCLAIMCLR, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCLAIMCLR = X[t, 64];
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        TRCCLAIMCLR = X[t, 64];

```

A.2.8.57 TRCAUTHSTATUS, Authentication Status Register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for debug.

Configurations

AArch64 register TRCAUTHSTATUS bits [31:0] are architecturally mapped to External register [B.2.2.7.62 TRCAUTHSTATUS, Authentication Status Register](#) on page 2085 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx00	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-436: AARCH64_TRCAUTHSTATUS bit assignments

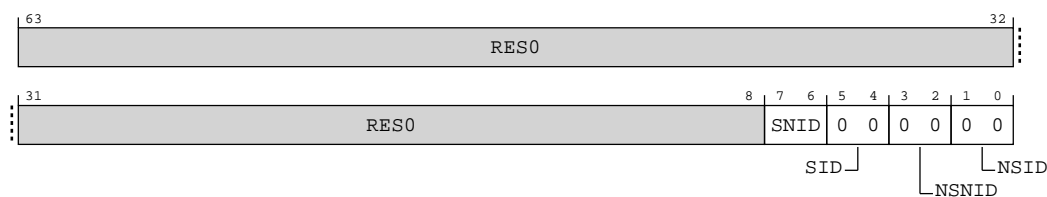


Table A-1119: TRCAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure Non-invasive Debug. Indicates whether Secure non-invasive debug features are implemented and enabled. 0b10 Secure Non-invasive Debug implemented and disabled. 0b11 Secure Non-invasive Debug implemented and enabled.	xx
[5:4]	SID	Secure Invasive Debug. Indicates whether Secure invasive debug features are implemented and enabled. 0b00 Secure invasive debug features not implemented.	0b00
[3:2]	NSNID	Non-secure Non-invasive Debug. Indicates whether Non-secure non-invasive debug features are implemented and enabled. 0b00 Non-secure non-invasive debug features not implemented.	0b00
[1:0]	NSID	Non-secure Invasive Debug. Indicates whether Non-secure invasive debug features are implemented and enabled. 0b00 Non-secure invasive debug features not implemented.	0b00

Access

MRS <Xt>, TRCAUTHSTATUS

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1110	0b110

Accessibility

MRS <Xt>, TRCAUTHSTATUS

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCAUTHSTATUS;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCAUTHSTATUS;
```

A.2.8.58 TRCDEVARCH, Device Architecture Register

Provides discovery information for the component.

Configurations

AArch64 register TRCDEVARCH bits [31:0] are architecturally mapped to External register [B.2.2.7.63 TRCDEVARCH, Device Architecture Register](#) on page 2086 bits [31:0].

Attributes

Width

64

Functional group

Trace unit registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0100	0111	0111	0101	0100	1010	0001	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-437: AARCH64_TRCDEVARCH bit assignments

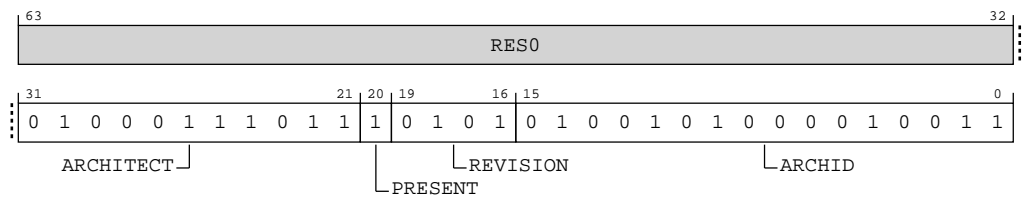


Table A-1121: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:21]	ARCHITECT	Architect. Defines the architect of the component. 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. 0b0101 ETMv4.5.	0b0101
[15:0]	ARCHID	Architecture ID. 0b0100101000010011 Arm PE Trace architecture ETMv4.	0x4A13

Access
MRS <Xt>, TRCDEVARCH

op0	op1	CRn	CRm	op2
0b10	0b001	0b0111	0b1111	0b110

Accessibility
MRS <Xt>, TRCDEVARCH

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = TRCDEVARCH;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

X[t, 64] = TRCDEVARCH;

A.2.9 AArch64 Special purpose register description

This section includes the register descriptions for all Special purpose registers in the Cortex®-R82 processor.

A.2.9.1 SPSR_EL1, Saved Program Status Register (EL1)

Holds the saved process state when an exception is taken to EL1.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0	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

An exception return from EL1 using AArch64 makes SPSR_EL1 become **UNKNOWN**.

Figure A-438: AARCH64_SPSR_EL1 bit assignments

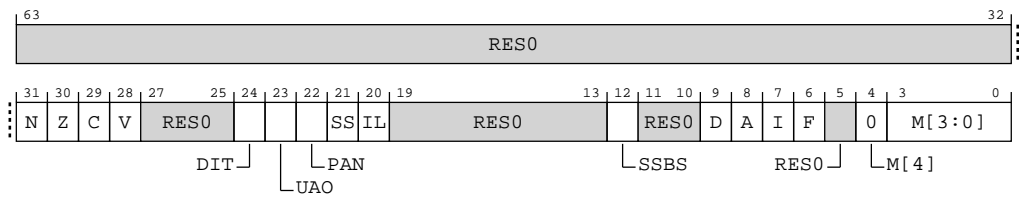


Table A-1123: SPSR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	N	Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL1, and copied to PSTATE.N on executing an exception return operation in EL1.	x
[30]	Z	Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL1, and copied to PSTATE.Z on executing an exception return operation in EL1.	x
[29]	C	Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL1, and copied to PSTATE.C on executing an exception return operation in EL1.	x
[28]	V	Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL1, and copied to PSTATE.V on executing an exception return operation in EL1.	x
[27:25]	RES0	Reserved	RES0
[24]	DIT	Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL1, and copied to PSTATE.DIT on executing an exception return operation in EL1.	x
[23]	UAO	User Access Override. Set to the value of PSTATE.UAO on taking an exception to EL1, and copied to PSTATE.UAO on executing an exception return operation in EL1.	x
[22]	PAN	Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL1, and copied to PSTATE.PAN on executing an exception return operation in EL1.	x
[21]	SS	Software Step. Set to the value of PSTATE.SS on taking an exception to EL1, and conditionally copied to PSTATE.SS on executing an exception return operation in EL1.	x
[20]	IL	Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL1, and copied to PSTATE.IL on executing an exception return operation in EL1.	x
[19:13]	RES0	Reserved	RES0
[12]	SSBS	Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL1, and copied to PSTATE.SSBS on executing an exception return operation in EL1.	x
[11:10]	RES0	Reserved	RES0
[9]	D	Debug exception mask. Set to the value of PSTATE.D on taking an exception to EL1, and copied to PSTATE.D on executing an exception return operation in EL1.	x
[8]	A	SError interrupt mask. Set to the value of PSTATE.A on taking an exception to EL1, and copied to PSTATE.A on executing an exception return operation in EL1.	x
[7]	I	IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL1, and copied to PSTATE.I on executing an exception return operation in EL1.	x
[6]	F	FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL1, and copied to PSTATE.F on executing an exception return operation in EL1.	x
[5]	RES0	Reserved	RES0
[4]	M[4]	Execution state. Set to 0b0, the value of PSTATE.nRW, on taking an exception to EL1 from AArch64 state, and copied to PSTATE.nRW on executing an exception return operation in EL1. 0b0 AArch64 execution state.	0b0

Bits	Name	Description	Reset
[3:0]	M[3:0]	<p>AArch64 Exception level and selected Stack Pointer.</p> <p>0b0000 EL0t.</p> <p>0b0100 EL1t.</p> <p>0b0101 EL1h.</p> <p>Other values are reserved. If SPSR_EL1.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL1 is an illegal return event, as described in <i>Illegal return events from AArch64 state</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>The bits in this field are interpreted as follows:</p> <ul style="list-style-type: none"> M[3:2]: On an exception to EL1: <ul style="list-style-type: none"> If the Effective value of AArch64-HCR_EL2.{NV, NV1} != {1,0} or the exception is not taken from EL1, then M[3:2] is set to the value of PSTATE.EL on taking an exception to EL1. If the Effective value of AArch64-HCR_EL2.{NV, NV1} == {1,0} and the exception is not taken from EL1, then M[3:2] is set to 0b10. M[3:2] is copied to PSTATE.EL on executing a legal exception return operation in EL1. M[1] is unused and is 0 for all non-reserved values. M[0] is set to the value of PSTATE.SP on taking an exception to EL1 and copied to PSTATE.SP on executing an exception return operation in EL1. 	xxxx

Access

MRS <Xt>, SPSR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0000	0b000

MSR SPSR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0000	0b000

Accessibility

MRS <Xt>, SPSR_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    X[t, 64] = SPSR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = SPSR_EL1;

```

MSR SPSR_EL1, <Xt>

```

if PSTATE.EL == EL0 then

```

```
    UNDEFINED;
    elsif PSTATE.EL == EL1 then
        SPSR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        SPSR_EL1 = X[t, 64];
```

A.2.9.2 ELR_EL1, Exception Link Register (EL1)

When taking an exception to EL1, holds the address to return to.

Configurations

This register is available in all configurations.

Attributes

Width

64

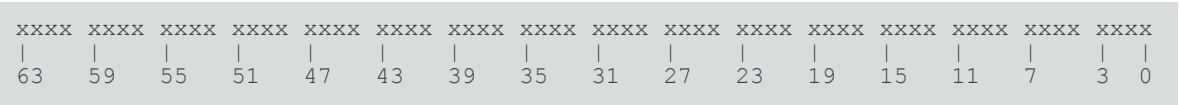
Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-439: AARCH64_EL1 bit assignments

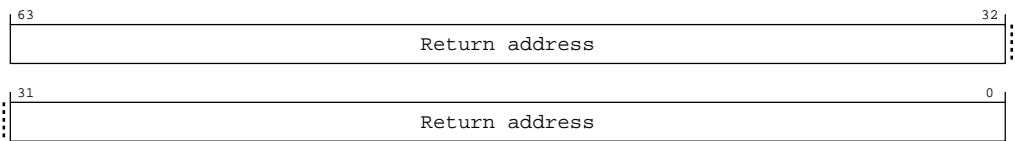


Table A-1126: ELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Return address. An exception return from EL1 using AArch64 makes ELR_EL1 become UNKNOWN .	64 { x }

Access

MRS <Xt>, ELR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0000	0b001

MSR ELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0000	0b001

Accessibility

MRS <Xt>, ELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    X[t, 64] = ELR_EL1;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ELR_EL1;
```

MSR ELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    ELR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    ELR_EL1 = X[t, 64];
```

A.2.9.3 SP_ELO, Stack Pointer (ELO)

Holds the stack pointer associated with EL0. At higher Exception levels, this is used as the current stack pointer when the value of AArch64-SPSel.SP is 0.

Configurations

This register is available in all configurations.

Attributes

Width

64

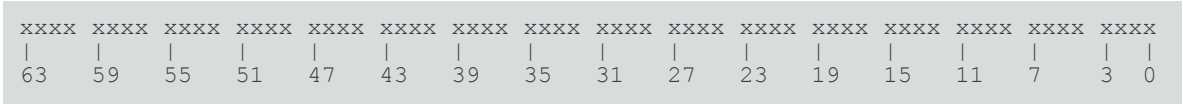
Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-440: AARCH64_SP_ELO bit assignments

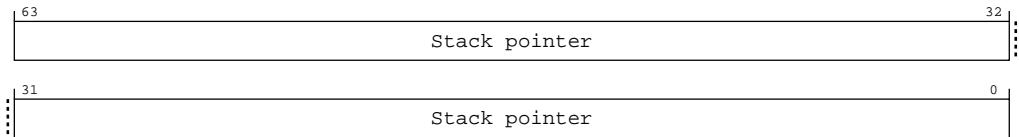


Table A-1129: SP_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Stack pointer.	64 {x}

Access

When the value of PSTATE.SP is 0, this register is accessible at all Exception levels as the current stack pointer.

MRS <Xt>, SP_ELO

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0001	0b000

MSR SP_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0001	0b000

Accessibility

When the value of PSTATE.SP is 0, this register is accessible at all Exception levels as the current stack pointer.

MRS <Xt>, SP_ELO

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if PSTATE.SP == '0' then
```

```
        UNDEFINED;
    else
        X[t, 64] = SP_ELO;
    elsif PSTATE.EL == EL2 then
        if PSTATE.SP == '0' then
            UNDEFINED;
        else
            X[t, 64] = SP_ELO;
```

MSR SP_ELO, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if PSTATE.SP == '0' then
        UNDEFINED;
    else
        SP_ELO = X[t, 64];
elsif PSTATE.EL == EL2 then
    if PSTATE.SP == '0' then
        UNDEFINED;
    else
        SP_ELO = X[t, 64];
```

A.2.9.4 DSPSR_ELO, Debug Saved Program Status Register

Holds the saved process state for Debug state. On entering Debug state, PSTATE information is written to this register. On exiting Debug state, values are copied from this register to PSTATE.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0	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-441: AARCH64_DSPSR_ELO bit assignments

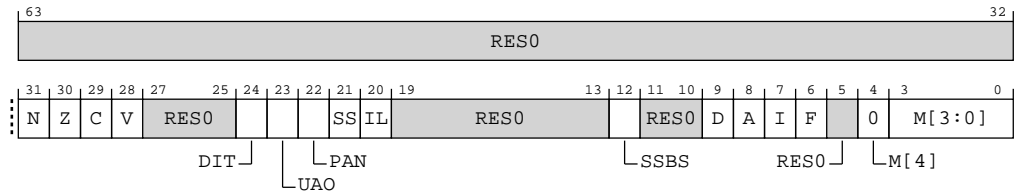


Table A-1132: DSPSR_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	N	Negative Condition flag. Set to the value of PSTATE.N on entering Debug state, and copied to PSTATE.N on exiting Debug state.	x
[30]	Z	Zero Condition flag. Set to the value of PSTATE.Z on entering Debug state, and copied to PSTATE.Z on exiting Debug state.	x
[29]	C	Carry Condition flag. Set to the value of PSTATE.C on entering Debug state, and copied to PSTATE.C on exiting Debug state.	x
[28]	V	Overflow Condition flag. Set to the value of PSTATE.V on entering Debug state, and copied to PSTATE.V on exiting Debug state.	x
[27:25]	RES0	Reserved	RES0
[24]	DIT	Data Independent Timing. Set to the value of PSTATE.DIT on entering Debug state, and copied to PSTATE.DIT on exiting Debug state.	x
[23]	UAO	User Access Override. Set to the value of PSTATE.UAO on entering Debug state, and copied to PSTATE.UAO on exiting Debug state.	x
[22]	PAN	Privileged Access Never. Set to the value of PSTATE.PAN on entering Debug state, and copied to PSTATE.PAN on exiting Debug state.	x
[21]	SS	Software Step. Set to the value of PSTATE.SS on entering Debug state, and conditionally copied to PSTATE.SS on exiting Debug state.	x
[20]	IL	Illegal Execution state. Set to the value of PSTATE.IL on entering Debug state, and copied to PSTATE.IL on exiting Debug state.	x
[19:13]	RES0	Reserved	RES0
[12]	SSBS	Speculative Store Bypass. Set to the value of PSTATE.SSBS on entering Debug state, and copied to PSTATE.SSBS on exiting Debug state.	x
[11:10]	RES0	Reserved	RES0
[9]	D	Debug exception mask. Set to the value of PSTATE.D on entering Debug state, and copied to PSTATE.D on exiting Debug state.	x
[8]	A	SError interrupt mask. Set to the value of PSTATE.A on entering Debug state, and copied to PSTATE.A on exiting Debug state.	x
[7]	I	IRQ interrupt mask. Set to the value of PSTATE.I on entering Debug state, and copied to PSTATE.I on exiting Debug state.	x
[6]	F	FIQ interrupt mask. Set to the value of PSTATE.F on entering Debug state, and copied to PSTATE.F on exiting Debug state.	x
[5]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[4]	M[4]	Execution state. Set to 0b0, the value of PSTATE.nRW, on entering Debug state from AArch64 state, and copied to PSTATE.nRW on exiting Debug state. 0b0 AArch64 execution state.	0b0
[3:0]	M[3:0]	AArch64 Exception level and selected Stack Pointer. 0b0000 EL0t. 0b0100 EL1t. 0b0101 EL1h. 0b1000 EL2t. 0b1001 EL2h. Other values are reserved. If DSPSR_ELO.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, exiting Debug state is an illegal return event, as described in <i>Illegal return events from AArch64 state</i> in the Arm® Architecture Reference Manual for A-profile architecture . The bits in this field are interpreted as follows: <ul style="list-style-type: none"> M[3:2] is set to the value of PSTATE.EL on entering Debug state and copied to PSTATE.EL on exiting Debug state. M[1] is unused and is 0 for all non-reserved values. M[0] is set to the value of PSTATE.SP on entering Debug state and copied to PSTATE.SP on exiting Debug state. 	xxxx

Access

MRS <Xt>, DSPSR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b000

MSR DSPSR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b000

Accessibility

MRS <Xt>, DSPSR_ELO

```
if !Halted() then
    UNDEFINED;
else
    X[t, 64] = DSPSR_ELO;
```

MSR DSPSR_ELO, <Xt>

```
if !Halted() then
    UNDEFINED;
else
    DSPSR_ELO = X[t, 64];
```

A.2.9.5 DLR_ELO, Debug Link Register

In Debug state, holds the address to restart from.

Configurations

This register is available in all configurations.

Attributes

Width

64

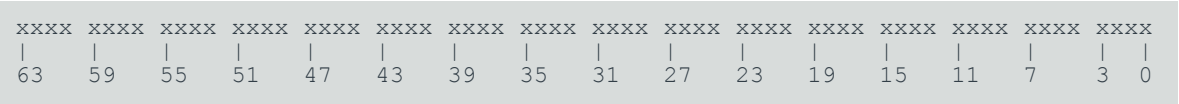
Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-442: AARCH64_DLR_ELO bit assignments

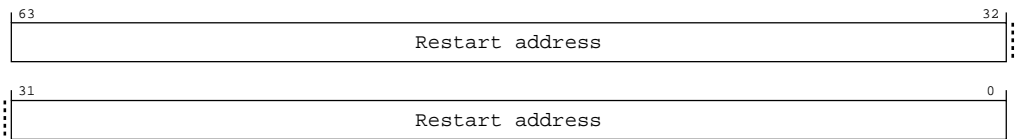


Table A-1135: DLR_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Restart address.	64 {x}

Access

MRS <Xt>, DLR_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b001

MSR DLR_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b001

Accessibility

MRS <Xt>, DLR_ELO

```
if !Halted() then
    UNDEFINED;
else
    X[t, 64] = DLR_ELO;
```

MSR DLR_ELO, <Xt>

```
if !Halted() then
    UNDEFINED;
else
    DLR_ELO = X[t, 64];
```

A.2.9.6 SPSR_EL2, Saved Program Status Register (EL2)

Holds the saved process state when an exception is taken to EL2.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0	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

An exception return from EL2 using AArch64 makes SPSR_EL2 become **UNKNOWN**.

Figure A-443: AARCH64_SPSR_EL2 bit assignments

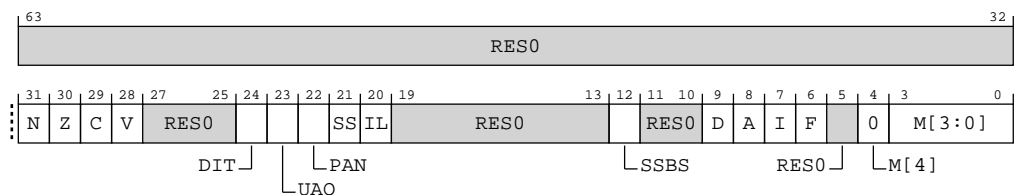


Table A-1138: SPSR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	N	Negative Condition flag. Set to the value of PSTATE.N on taking an exception to EL2, and copied to PSTATE.N on executing an exception return operation in EL2.	x
[30]	Z	Zero Condition flag. Set to the value of PSTATE.Z on taking an exception to EL2, and copied to PSTATE.Z on executing an exception return operation in EL2.	x
[29]	C	Carry Condition flag. Set to the value of PSTATE.C on taking an exception to EL2, and copied to PSTATE.C on executing an exception return operation in EL2.	x
[28]	V	Overflow Condition flag. Set to the value of PSTATE.V on taking an exception to EL2, and copied to PSTATE.V on executing an exception return operation in EL2.	x
[27:25]	RES0	Reserved	RES0
[24]	DIT	Data Independent Timing. Set to the value of PSTATE.DIT on taking an exception to EL2, and copied to PSTATE.DIT on executing an exception return operation in EL2.	x
[23]	UAO	User Access Override. Set to the value of PSTATE.UAO on taking an exception to EL2, and copied to PSTATE.UAO on executing an exception return operation in EL2.	x
[22]	PAN	Privileged Access Never. Set to the value of PSTATE.PAN on taking an exception to EL2, and copied to PSTATE.PAN on executing an exception return operation in EL2.	x
[21]	SS	Software Step. Set to the value of PSTATE.SS on taking an exception to EL2, and conditionally copied to PSTATE.SS on executing an exception return operation in EL2.	x
[20]	IL	Illegal Execution state. Set to the value of PSTATE.IL on taking an exception to EL2, and copied to PSTATE.IL on executing an exception return operation in EL2.	x
[19:13]	RES0	Reserved	RES0
[12]	SSBS	Speculative Store Bypass. Set to the value of PSTATE.SSBS on taking an exception to EL2, and copied to PSTATE.SSBS on executing an exception return operation in EL2.	x
[11:10]	RES0	Reserved	RES0
[9]	D	Debug exception mask. Set to the value of PSTATE.D on taking an exception to EL2, and copied to PSTATE.D on executing an exception return operation in EL2.	x

Bits	Name	Description	Reset
[8]	A	SError interrupt mask. Set to the value of PSTATE.A on taking an exception to EL2, and copied to PSTATE.A on executing an exception return operation in EL2.	x
[7]	I	IRQ interrupt mask. Set to the value of PSTATE.I on taking an exception to EL2, and copied to PSTATE.I on executing an exception return operation in EL2.	x
[6]	F	FIQ interrupt mask. Set to the value of PSTATE.F on taking an exception to EL2, and copied to PSTATE.F on executing an exception return operation in EL2.	x
[5]	RES0	Reserved	RES0
[4]	M[4]	Execution state. Set to 0b0, the value of PSTATE.nRW, on taking an exception to EL2 from AArch64 state, and copied to PSTATE.nRW on executing an exception return operation in EL2. 0b0 AArch64 execution state.	0b0
[3:0]	M[3:0]	AArch64 Exception level and selected Stack Pointer. 0b0000 EL0t. 0b0100 EL1t. 0b0101 EL1h. 0b1000 EL2t. 0b1001 EL2h. Other values are reserved. If SPSR_EL2.M[3:0] has a Reserved value, or a value for an unimplemented Exception level, executing an exception return operation in EL2 is an illegal return event, as described in <i>Illegal return events from AArch64 state</i> in the Arm® Architecture Reference Manual for A-profile architecture . The bits in this field are interpreted as follows: <ul style="list-style-type: none"> M[3:2] is set to the value of PSTATE.EL on taking an exception to EL2 and copied to PSTATE.EL on executing an exception return operation in EL2. M[1] is unused and is 0 for all non-reserved values. M[0] is set to the value of PSTATE.SP on taking an exception to EL2 and copied to PSTATE.SP on executing an exception return operation in EL2. 	xxxx

Access

MRS <Xt>, SPSR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b000

MSR SPSR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b000

Accessibility

MRS <Xt>, SPSR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = SPSR_EL2;
```

MSR SPSR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    SPSR_EL2 = X[t, 64];
```

A.2.9.7 ELR_EL2, Exception Link Register (EL2)

When taking an exception to EL2, holds the address to return to.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-444: AARCH64_ELR_EL2 bit assignments

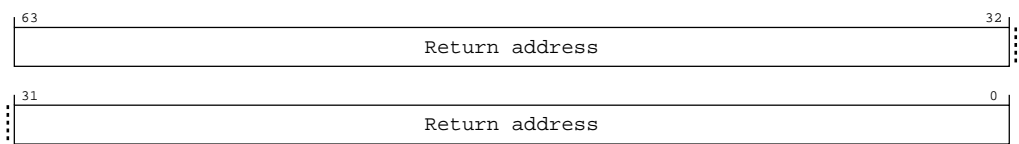


Table A-1141: ELR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Return address. An exception return from EL2 using AArch64 makes ELR_EL2 become UNKNOWN .	64 {x}

Access

MRS <Xt>, ELR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b001

MSR ELR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0000	0b001

Accessibility

MRS <Xt>, ELR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = ELR_EL2;
```

MSR ELR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    ELR_EL2 = X[t, 64];
```

A.2.9.8 SP_EL1, Stack Pointer (EL1)

Holds the stack pointer associated with EL1. When executing at EL1, the value of AArch64-SPSel.SP determines the current stack pointer:

- When AArch64-SPSel.SP == 0b0, current stack pointer is AArch64-SP_EL0
- When AArch64-SPSel.SP == 0b1, current stack pointer is AArch64-SP_EL1

Configurations

This register is available in all configurations.

Attributes

Width

64

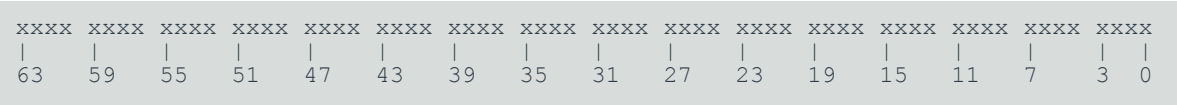
Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-445: AARCH64_SP_EL1 bit assignments

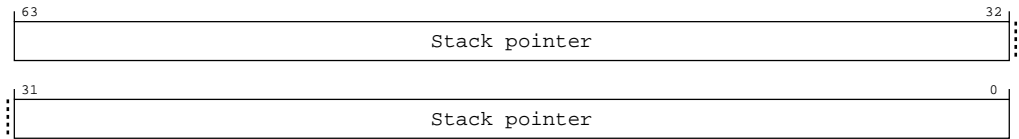


Table A-1144: SP_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Stack pointer.	64 {x}

Access

This accessibility information only applies to accesses using the MRS or MSR instructions.

When the value of AArch64-SPSel.SP is 1, this register is also accessible at EL1 as the current stack pointer.



When the value of AArch64-SPSel.SP is 0, AArch64-SP_ELO is used as the current stack pointer at all Exception levels.

MRS <Xt>, SP_EL1

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0001	0b000

MSR SP_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0001	0b000

Accessibility

This accessibility information only applies to accesses using the MRS or MSR instructions.

When the value of AArch64-SPSel.SP is 1, this register is also accessible at EL1 as the current stack pointer.



When the value of AArch64-SPSel.SP is 0, AArch64-SP_ELO is used as the current stack pointer at all Exception levels.

MRS <Xt>, SP_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SP_EL1;
```

MSR SP_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    SP_EL1 = X[t, 64];
```

A.2.9.9 SPSR_irq, Saved Program Status Register (IRQ mode)

Holds the saved process state when an exception is taken to IRQ mode.

Configurations

This register is available in all configurations.

Attributes

Width

64

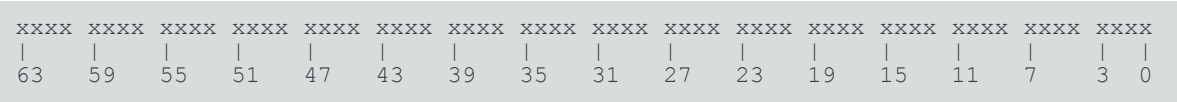
Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-446: AARCH64_SPSR_IRQ bit assignments

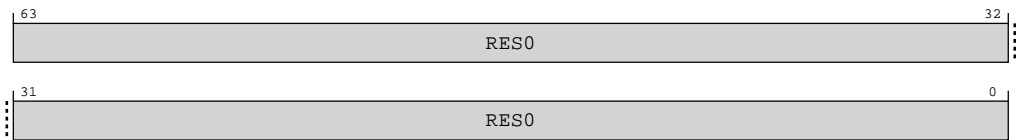


Table A-1147: SPSR_irq bit descriptions

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

Access

MRS <Xt>, SPSR_irq

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b000

MSR SPSR_irq, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b000

Accessibility

MRS <Xt>, SPSR_irq

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

MSR SPSR_irq, <Xt>

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

A.2.9.10 SPSR_abt, Saved Program Status Register (Abort mode)

Holds the saved process state when an exception is taken to Abort mode.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-447: AARCH64_SPSR_ABT bit assignments

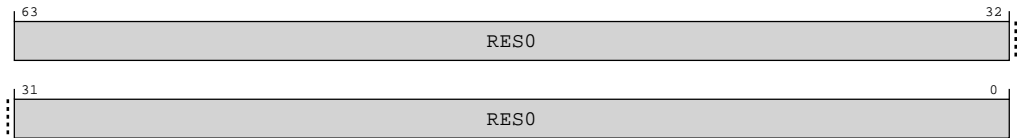


Table A-1150: SPSR_abt bit descriptions

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

Access

MRS <Xt>, SPSR_abt

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b001

MSR SPSR_abt, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b001

Accessibility

MRS <Xt>, SPSR_abt

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = SPSR_abt;
```

MSR SPSR_abt, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    SPSR_abt = X[t, 64];
```


A.2.9.11 SPSR_und, Saved Program Status Register (Undefined mode)

Holds the saved process state when an exception is taken to Undefined mode.

Configurations

This register is available in all configurations.

Attributes

Width

64

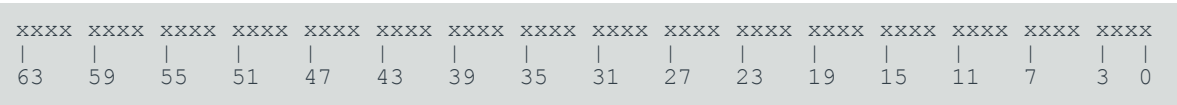
Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-448: AARCH64_SPSR_UND bit assignments

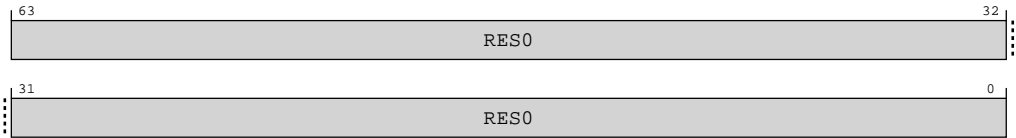


Table A-1153: SPSR_und bit descriptions

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

Access

MRS <Xt>, SPSR_und

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b010

MSR SPSR_und, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b010

Accessibility

MRS <Xt>, SPSR_und

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

MSR SPSR_und, <Xt>

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

A.2.9.12 SPSR_fiq, Saved Program Status Register (FIQ mode)

Holds the saved process state when an exception is taken to FIQ mode.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Special-purpose registers

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	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-449: AARCH64_SPSR_FIQ bit assignments

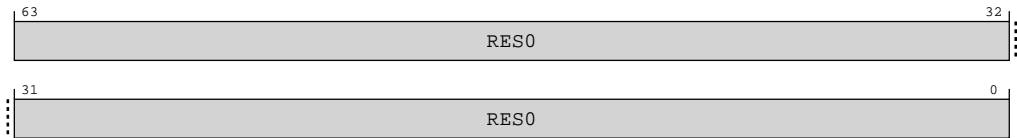


Table A-1156: SPSR_fiq bit descriptions

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

Access

MRS <Xt>, SPSR_fiq

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b011

MSR SPSR_fiq, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0100	0b0011	0b011

Accessibility

MRS <Xt>, SPSR_fiq

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = SPSR_fiq;
```

MSR SPSR_fiq, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    SPSR_fiq = X[t, 64];
```

A.2.10 AArch64 Generic Timer register description

This section includes the register descriptions for all Generic Timer registers in the Cortex®-R82 processor.

A.2.10.1 CNTKCTL_EL1, Counter-timer Kernel Control Register

This register controls the generation of an event stream from the virtual counter, and access from EL0 to the physical counter, virtual counter, EL1 physical timers, and the virtual timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-450: AARCH64_CNTKCTL_EL1 bit assignments

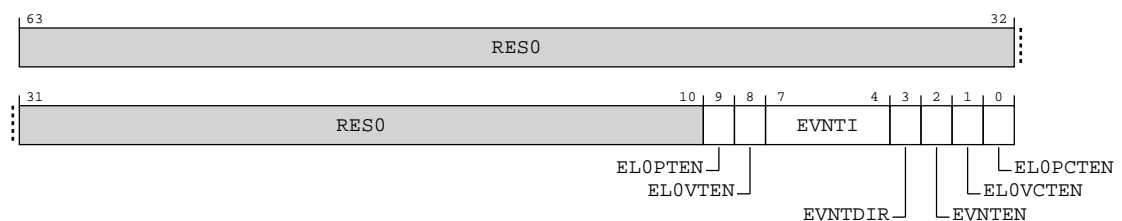


Table A-1159: CNTKCTL_EL1 bit descriptions

Bits	Name	Description	Reset
[63:10]	RES0	Reserved	RES0
[9]	ELOPTEN	<p>Traps ELO accesses to the physical timer registers to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, the following registers are trapped, reported using EC syndrome value 0x18: <ul style="list-style-type: none"> AArch64-CNTP_CTL_ELO, AArch64-CNTP_CVAL_ELO, and AArch64-CNTP_TVAL_ELO. <p>0b0 ELO accesses to the physical timer registers are trapped to EL1.</p> <p>0b1 This control does not cause any instructions to be trapped.</p>	x
[8]	ELOVTEN	<p>Traps ELO accesses to the virtual timer registers to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to the following registers are trapped, reported using EC syndrome value 0x18: <ul style="list-style-type: none"> AArch64-CNTV_CTL_ELO, AArch64-CNTV_CVAL_ELO, and AArch64-CNTV_TVAL_ELO. <p>0b0 ELO accesses to the virtual timer registers are trapped.</p> <p>0b1 This control does not cause any instructions to be trapped.</p>	x
[7:4]	EVNTI	<p>Selects which bit of AArch64-CNTVCT_ELO, as seen from EL1, is the trigger for the event stream generated from that counter when that stream is enabled.</p> <p>This field selects a trigger bit in the range 0 to 15 of AArch64-CNTVCT_ELO.</p>	xxxx
[3]	EVNTDIR	<p>Controls which transition of the AArch64-CNTVCT_ELO trigger bit, as seen from EL1 and defined by EVNTI, generates an event when the event stream is enabled.</p> <p>0b0 A 0 to 1 transition of the trigger bit triggers an event.</p> <p>0b1 A 1 to 0 transition of the trigger bit triggers an event.</p>	x
[2]	EVNTEN	<p>Enables the generation of an event stream from AArch64-CNTVCT_ELO as seen from EL1.</p> <p>0b0 Disables the event stream.</p> <p>0b1 Enables the event stream.</p>	x
[1]	ELOVCTEN	<p>Traps ELO accesses to the frequency register and virtual counter register to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, as follows:</p> <ul style="list-style-type: none"> In AArch64 state, accesses to the following registers are trapped and reported using EC syndrome value 0x18: <ul style="list-style-type: none"> AArch64-CNTVCT_ELO and if AArch64-CNTKCTL_EL1.ELOPCTEN is 0, AArch64-CNTFRQ_ELO. <p>0b0 ELO accesses to the frequency register and virtual counter registers are trapped.</p> <p>0b1 This control does not cause any instructions to be trapped.</p>	x

Bits	Name	Description	Reset
[0]	ELOPCTEN	Traps ELO accesses to the frequency register and physical counter register to EL1, or to EL2 when it is implemented and enabled for the current Security state and AArch64-HCR_EL2.TGE is 1, as follows: <ul style="list-style-type: none">In AArch64 state, the following registers are trapped, reported using EC syndrome value 0x18:<ul style="list-style-type: none">AArch64-CNTPCT_ELO and if AArch64-CNTKCTL_EL1.ELOVCTEN is 0, AArch64-CNTFRQ_ELO. <p>0b0</p> ELO accesses to the frequency register and physical counter register are trapped. <p>0b1</p> This control does not cause any instructions to be trapped.	x

Access

MRS <Xt>, CNTKCTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b11110	0b0001	0b000

MSR CNTKCTL_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b11110	0b0001	0b000

Accessibility

MRS <Xt>, CNTKCTL_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    X[t, 64] = CNTKCTL_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CNTKCTL_EL1;
```

MSR CNTKCTL_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    CNTKCTL_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    CNTKCTL_EL1 = X[t, 64];
```

A.2.10.2 CNTFRQ_ELO, Counter-timer Frequency Register

This register is provided so that software can discover the frequency of the system counter. It must be programmed with this value as part of system initialization. The value of the register is not interpreted by hardware.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-451: AARCH64_CNTFRQ_ELO bit assignments

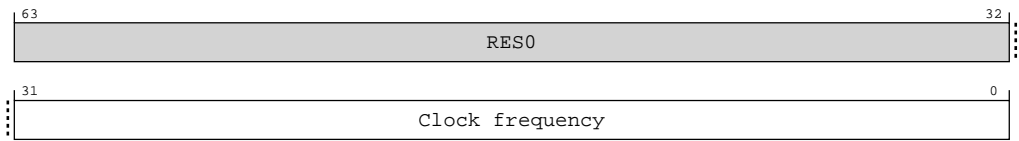


Table A-1162: CNTFRQ_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	None	Clock frequency. Indicates the system counter clock frequency, in Hz.	32 {x}

Access

MRS <Xt>, CNTFRQ_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	0b0000	0b000

MSR CNTFRQ_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	0b0000	0b000

Accessibility

MRS <Xt>, CNTFRQ_ELO

```
if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.<EL0PCTEN,EL0VCTEN> == '00' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            X[t, 64] = CNTFRQ_ELO;
    elseif PSTATE.EL == EL1 then
        X[t, 64] = CNTFRQ_ELO;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = CNTFRQ_ELO;
```

MSR CNTFRQ_ELO, <Xt>

```
if IsHighestEL(PSTATE.EL) then
    CNTFRQ_ELO = X[t, 64];
else
    UNDEFINED;
```

A.2.10.3 CNTPCT_ELO, Counter-timer Physical Count Register

Holds the 64-bit physical count value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

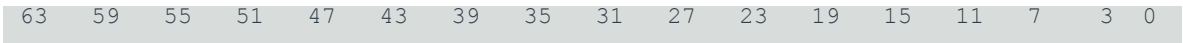
Generic Timer registers

Access type

See bit descriptions

Reset value

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-452: AARCH64_Cntpct_El0 bit assignments

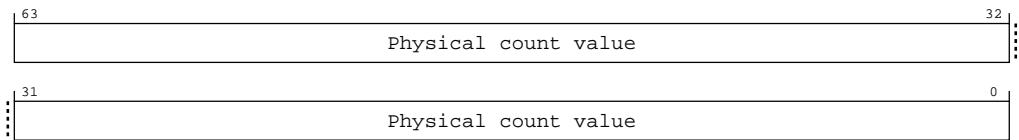


Table A-1165: Cntpct_El0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Physical count value.	64 {x}

Access

MRS <Xt>, Cntpct_El0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0000	0b001

Accessibility

MRS <Xt>, Cntpct_El0

```
if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0PCTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CNTHCTL_EL2.EL1PCTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PhysicalCountInt();
elseif PSTATE.EL == EL1 then
    if CNTHCTL_EL2.EL1PCTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = PhysicalCountInt();
elseif PSTATE.EL == EL2 then
    X[t, 64] = PhysicalCountInt();
```

A.2.10.4 CNTVCT_EL0, Counter-timer Virtual Count Register

Holds the 64-bit virtual count value. The virtual count value is equal to the physical count value minus the virtual offset visible in AArch64-CNTVOFF_EL2.

Configurations

This register is available in all configurations.

Attributes

Width

64

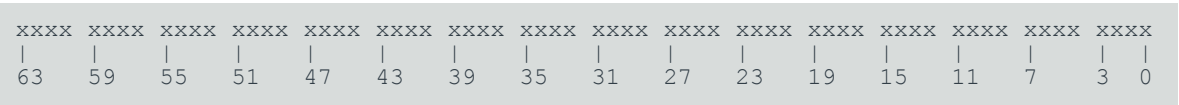
Functional group

Generic Timer registers

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-453: AARCH64_CNTVCT_EL0 bit assignments

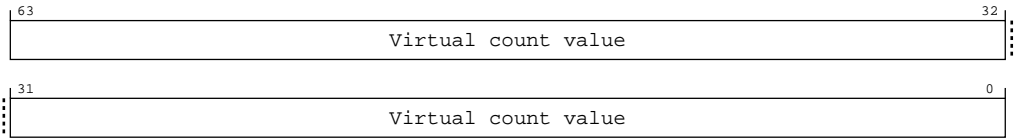


Table A-1167: CNTVCT_EL0 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Virtual count value.	64 {x}

Access

MRS <Xt>, CNTVCT_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	0b0000	0b010

Accessibility

MRS <Xt>, CNTVCT_ELO

```
if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0VCTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            X[t, 64] = PhysicalCountInt() - CNTVOFF_EL2;
    elseif PSTATE.EL == EL1 then
        X[t, 64] = PhysicalCountInt() - CNTVOFF_EL2;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = PhysicalCountInt() - CNTVOFF_EL2;
```

A.2.10.5 CNTP_TVAL_ELO, Counter-timer Physical Timer TimerValue Register

Holds the timer value for the EL1 physical timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-454: AARCH64_CNTP_TVAL_ELO bit assignments

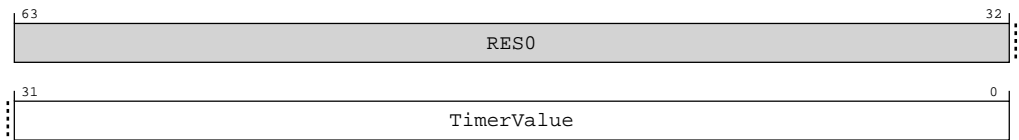


Table A-1169: CNTP_TVAL_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	TimerValue	<p>The TimerValue view of the EL1 physical timer.</p> <p>On a read of this register:</p> <ul style="list-style-type: none">If AArch64-CNTP_CTL_ELO.ENABLE is 0, the value returned is UNKNOWN.If AArch64-CNTP_CTL_ELO.ENABLE is 1, the value returned is (AArch64-CNTP_CVAL_ELO - AArch64-CNTPCT_ELO). <p>On a write of this register, AArch64-CNTP_CVAL_ELO is set to (AArch64-CNTPCT_ELO + TimerValue), where TimerValue is treated as a signed 32-bit integer.</p> <p>When AArch64-CNTP_CTL_ELO.ENABLE is 1, the timer condition is met when (AArch64-CNTPCT_ELO - AArch64-CNTP_CVAL_ELO) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:</p> <ul style="list-style-type: none">AArch64-CNTP_CTL_ELO.ISTATUS is set to 1.If AArch64-CNTP_CTL_ELO.IMASK is 0, an interrupt is generated. <p>When AArch64-CNTP_CTL_ELO.ENABLE is 0, the timer condition is not met, but AArch64-CNTPCT_ELO continues to count, so the TimerValue view appears to continue to count down.</p> <p>Note: The value of AArch64-CNTPCT_ELO used in these calculations is the value seen at the Exception Level that the AArch64-CNTPCT_ELO register is being read or written from.</p>	32 {x}

Access

MRS <Xt>, CNTP_TVAL_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

MSR CNTP_TVAL_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

Accessibility

MRS <Xt>, CNTP_TVAL_ELO

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0PTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            if CNTP_CTL_EL0.ENABLE == '0' then
                X[t, 64] = bits(64) UNKNOWN;
            else
                X[t, 64] = CNTP_CVAL_ELO - PhysicalCountInt();
            elsif PSTATE.EL == EL1 then
                if CNTHCTL_EL2.EL1PCEN == '0' then
                    AArch64.SystemAccessTrap(EL2, 0x18);
                else
                    if CNTP_CTL_EL0.ENABLE == '0' then
                        X[t, 64] = bits(64) UNKNOWN;
                    else
                        X[t, 64] = CNTP_CVAL_ELO - PhysicalCountInt();
            elsif PSTATE.EL == EL2 then
                if CNTP_CTL_EL0.ENABLE == '0' then
                    X[t, 64] = bits(64) UNKNOWN;
                else
                    X[t, 64] = CNTP_CVAL_ELO - PhysicalCountInt();

```

MSR CNTP_TVAL_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0PTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            CNTP_CVAL_ELO = SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt();
            elsif PSTATE.EL == EL1 then
                if CNTHCTL_EL2.EL1PCEN == '0' then
                    AArch64.SystemAccessTrap(EL2, 0x18);
                else
                    CNTP_CVAL_ELO = SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt();
            elsif PSTATE.EL == EL2 then
                CNTP_CVAL_ELO = SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt();

```

A.2.10.6 CNTP_CTL_ELO, Counter-timer Physical Timer Control Register

Control register for the EL1 physical timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-455: AARCH64_CNTP_CTL_EL0 bit assignments

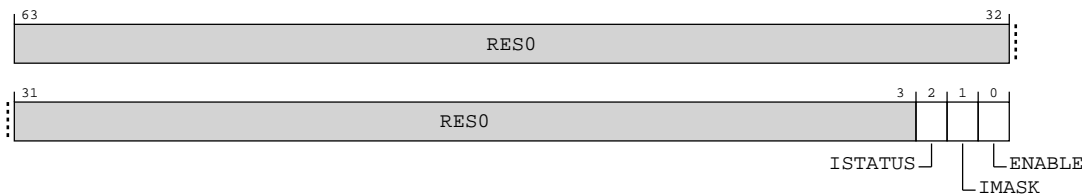


Table A-1172: CNTP_CTL_EL0 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2]	ISTATUS	<p>The status of the timer. This bit indicates whether the timer condition is met:</p> <p>0b0</p> <p>Timer condition is not met.</p> <p>0b1</p> <p>Timer condition is met.</p> <p>When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.</p> <p>When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.</p> <p>Access to this field is: RO</p>	x

Bits	Name	Description	Reset
[1]	IMASK	<p>Timer interrupt mask bit. Permitted values are:</p> <p>0b0 Timer interrupt is not masked by the IMASK bit.</p> <p>0b1 Timer interrupt is masked by the IMASK bit.</p> <p>For more information, see the description of the ISTATUS bit.</p>	x
[0]	ENABLE	<p>Enables the timer. Permitted values are:</p> <p>0b0 Timer disabled.</p> <p>0b1 Timer enabled.</p> <p>Setting this bit to 0 disables the timer output signal, but the timer value accessible from AArch64-CNTP_TVAL_ELO continues to count down.</p> <p>Note: Disabling the output signal might be a power-saving option.</p>	x

Access

MRS <Xt>, CNTP_CTL_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	0b0010	0b001

MSR CNTP_CTL_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	0b0010	0b001

Accessibility

MRS <Xt>, CNTP_CTL_ELO

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0PTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CNTP_CTL_ELO;
    elsif PSTATE.EL == EL1 then
        if CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CNTP_CTL_ELO;
    elsif PSTATE.EL == EL2 then
        X[t, 64] = CNTP_CTL_ELO;

```

MSR CNTP_CTL_EL0, <Xt>

```
if PSTATE.EL == EL0 then
  if CNTKCTL_EL1.EL0PTEN == '0' then
    if HCR_EL2.TGE == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      AArch64.SystemAccessTrap(EL1, 0x18);
    elsif CNTHCTL_EL2.EL1PCEN == '0' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      CNTP_CTL_EL0 = X[t, 64];
  elsif PSTATE.EL == EL1 then
    if CNTHCTL_EL2.EL1PCEN == '0' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      CNTP_CTL_EL0 = X[t, 64];
  elsif PSTATE.EL == EL2 then
    CNTP_CTL_EL0 = X[t, 64];
```

A.2.10.7 CNTP_CVAL_EL0, Counter-timer Physical Timer CompareValue Register

Holds the compare value for the EL1 physical timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-456: AARCH64_CNTP_CVAL_ELO bit assignments

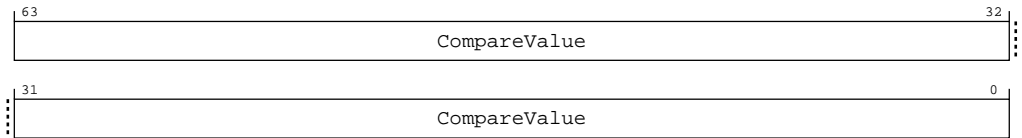


Table A-1175: CNTP_CVAL_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	CompareValue	<p>Holds the EL1 physical timer CompareValue.</p> <p>When AArch64-CNTP_CTL_ELO.ENABLE is 1, the timer condition is met when (AArch64-CNTPCT_ELO - CompareValue) is greater than or equal to zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met:</p> <ul style="list-style-type: none">AArch64-CNTP_CTL_ELO.ISTATUS is set to 1.If AArch64-CNTP_CTL_ELO.IMASK is 0, an interrupt is generated. <p>When AArch64-CNTP_CTL_ELO.ENABLE is 0, the timer condition is not met, but AArch64-CNTPCT_ELO continues to count.</p> <p>If the Generic counter is implemented at a size less than 64 bits, then this field is permitted to be implemented at the same width as the counter, and the upper bits are RES0.</p> <p>The value of this field is treated as zero-extended in all counter calculations.</p>	64 {x}

Access

MRS <Xt>, CNTP_CVAL_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b010

MSR CNTP_CVAL_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b010

Accessibility

MRS <Xt>, CNTP_CVAL_ELO

```
if PSTATE.EL == EL0 then
  if CNTKCTL_EL1.EL0PTEN == '0' then
    if HCR_EL2.TGE == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
      AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CNTHCTL_EL2.EL1PCEN == '0' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```

        X[t, 64] = CNTP_CVAL_EL0;
    elseif PSTATE.EL == EL1 then
        if CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            X[t, 64] = CNTP_CVAL_EL0;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = CNTP_CVAL_EL0;

```

MSR CNTP_CVAL_EL0, <Xt>

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0PTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        CNTP_CVAL_EL0 = X[t, 64];
elseif PSTATE.EL == EL1 then
    if CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        CNTP_CVAL_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    CNTP_CVAL_EL0 = X[t, 64];

```

A.2.10.8 CNTV_TVAL_EL0, Counter-timer Virtual Timer TimerValue Register

Holds the timer value for the EL1 virtual timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-457: AARCH64_CNTV_TVAL_ELO bit assignments

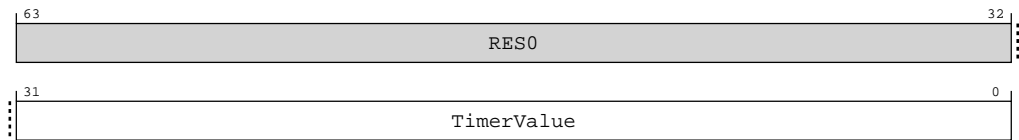


Table A-1178: CNTV_TVAL_ELO bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	TimerValue	<p>The TimerValue view of the EL1 virtual timer.</p> <p>On a read of this register:</p> <ul style="list-style-type: none">If AArch64-CNTV_CTL_ELO.ENABLE is 0, the value returned is UNKNOWN.If AArch64-CNTV_CTL_ELO.ENABLE is 1, the value returned is (AArch64-CNTV_CVAL_ELO - AArch64-CNTVCT_ELO). <p>On a write of this register, AArch64-CNTV_CVAL_ELO is set to (AArch64-CNTVCT_ELO + TimerValue), where TimerValue is treated as a signed 32-bit integer.</p> <p>When AArch64-CNTV_CTL_ELO.ENABLE is 1, the timer condition is met when (AArch64-CNTVCT_ELO - AArch64-CNTV_CVAL_ELO) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:</p> <ul style="list-style-type: none">AArch64-CNTV_CTL_ELO.ISTATUS is set to 1.If AArch64-CNTV_CTL_ELO.IMASK is 0, an interrupt is generated. <p>When AArch64-CNTV_CTL_ELO.ENABLE is 0, the timer condition is not met, but AArch64-CNTVCT_ELO continues to count, so the TimerValue view appears to continue to count down.</p>	32 {x}

Access

MRS <Xt>, CNTV_TVAL_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

MSR CNTV_TVAL_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

Accessibility

MRS <Xt>, CNTV_TVAL_ELO

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0VTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            if CNTV_CTL_EL0.ENABLE == '0' then
                X[t, 64] = bits(64) UNKNOWN;
            else
                X[t, 64] = CNTV_CVAL_ELO - (PhysicalCountInt() - CNTVOFF_EL2);
    elseif PSTATE.EL == EL1 then
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_ELO - (PhysicalCountInt() - CNTVOFF_EL2);
    elseif PSTATE.EL == EL2 then
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_ELO - (PhysicalCountInt() - CNTVOFF_EL2);

```

MSR CNTV_TVAL_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0VTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            CNTV_CVAL_ELO = (SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt()) -
                CNTVOFF_EL2;
    elseif PSTATE.EL == EL1 then
        CNTV_CVAL_ELO = (SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt()) -
            CNTVOFF_EL2;
    elseif PSTATE.EL == EL2 then
        CNTV_CVAL_ELO = (SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt()) -
            CNTVOFF_EL2;

```

A.2.10.9 CNTV_CTL_ELO, Counter-timer Virtual Timer Control Register

Control register for the virtual timer.

Configurations

This register is available in all configurations.

Attributes

Width

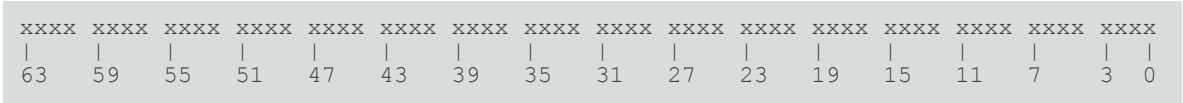
64

Functional group

Generic Timer registers

Access type
RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-458: AARCH64_CNTV_CTL_EL0 bit assignments

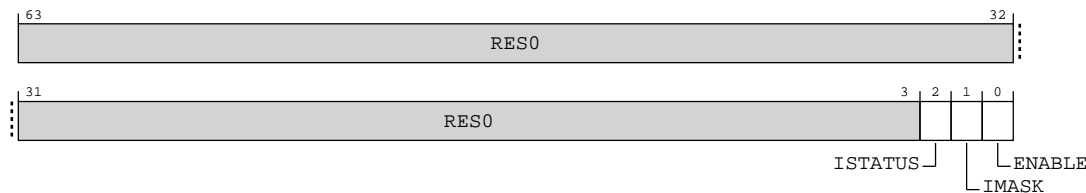


Table A-1181: CNTV_CTL_EL0 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0
[2]	ISTATUS	<p>The status of the timer. This bit indicates whether the timer condition is met:</p> <p>0b0</p> <p>Timer condition is not met.</p> <p>0b1</p> <p>Timer condition is met.</p> <p>When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.</p> <p>When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.</p> <p>Access to this field is: RO</p>	x
[1]	IMASK	<p>Timer interrupt mask bit. Permitted values are:</p> <p>0b0</p> <p>Timer interrupt is not masked by the IMASK bit.</p> <p>0b1</p> <p>Timer interrupt is masked by the IMASK bit.</p> <p>For more information, see the description of the ISTATUS bit.</p>	x

Bits	Name	Description	Reset
[0]	ENABLE	<p>Enables the timer. Permitted values are:</p> <p>0b0 Timer disabled.</p> <p>0b1 Timer enabled.</p> <p>Setting this bit to 0 disables the timer output signal, but the timer value accessible from AArch64-CNTV_TVAL_ELO continues to count down.</p> <p>Note: Disabling the output signal might be a power-saving option.</p>	x

Access

MRS <Xt>, CNTV_CTL_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	0b0011	0b001

MSR CNTV_CTL_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b11110	0b0011	0b001

Accessibility

MRS <Xt>, CNTV_CTL_ELO

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0VTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            X[t, 64] = CNTV_CTL_ELO;
    elseif PSTATE.EL == EL1 then
        X[t, 64] = CNTV_CTL_ELO;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = CNTV_CTL_ELO;

```

MSR CNTV_CTL_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0VTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            CNTV_CTL_ELO = X[t, 64];
    elseif PSTATE.EL == EL1 then
        CNTV_CTL_ELO = X[t, 64];

```

```
elseif PSTATE.EL == EL2 then
    CNTV_CTL_EL0 = X[t, 64];
```

A.2.10.10 CNTV_CVAL_EL0, Counter-timer Virtual Timer CompareValue Register

Holds the compare value for the virtual timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-459: AARCH64_CNTV_CVAL_EL0 bit assignments

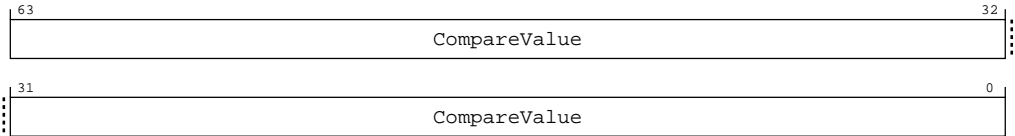


Table A-1184: CNTV_CVAL_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	CompareValue	<p>Holds the EL1 virtual timer CompareValue.</p> <p>When AArch64-CNTV_CTL_ELO.ENABLE is 1, the timer condition is met when (AArch64-CNTVCT_ELO - CompareValue) is greater than or equal to zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met:</p> <ul style="list-style-type: none"> AArch64-CNTV_CTL_ELO.ISTATUS is set to 1. If AArch64-CNTV_CTL_ELO.IMASK is 0, an interrupt is generated. <p>When AArch64-CNTV_CTL_ELO.ENABLE is 0, the timer condition is not met, but AArch64-CNTVCT_ELO continues to count.</p> <p>If the Generic counter is implemented at a size less than 64 bits, then this field is permitted to be implemented at the same width as the counter, and the upper bits are RES0.</p> <p>The value of this field is treated as zero-extended in all counter calculations.</p>	64 {x}

Access

MRS <Xt>, CNTV_CVAL_ELO

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b010

MSR CNTV_CVAL_ELO, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b010

Accessibility

MRS <Xt>, CNTV_CVAL_ELO

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0VTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            X[t, 64] = CNTV_CVAL_ELO;
    elseif PSTATE.EL == EL1 then
        X[t, 64] = CNTV_CVAL_ELO;
    elseif PSTATE.EL == EL2 then
        X[t, 64] = CNTV_CVAL_ELO;

```

MSR CNTV_CVAL_ELO, <Xt>

```

if PSTATE.EL == EL0 then
    if CNTKCTL_EL1.EL0VTEN == '0' then
        if HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);

```



```
else
    CNTV_CVAL_EL0 = X[t, 64];
elseif PSTATE.EL == EL1 then
    CNTV_CVAL_EL0 = X[t, 64];
elseif PSTATE.EL == EL2 then
    CNTV_CVAL_EL0 = X[t, 64];
```

A.2.10.11 CNTVOFF_EL2, Counter-timer Virtual Offset Register

Holds the 64-bit virtual offset. This is the offset between the physical count value visible in AArch64-CNTPCT_ELO and the virtual count value visible in AArch64-CNTVCT_ELO.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-460: AARCH64_CNTVOFF_EL2 bit assignments

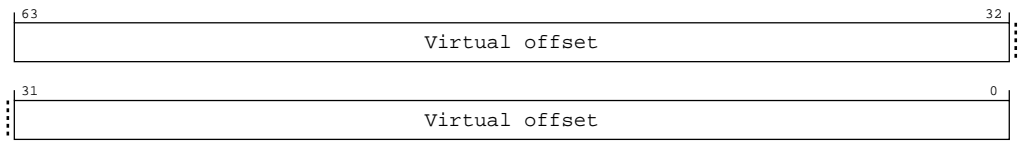


Table A-1187: CNTVOFF_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	None	Virtual offset. If the Generic counter is implemented at a size less than 64 bits, then this field is permitted to be implemented at the same width as the counter, and the upper bits are RES0 . The value of this field is treated as zero-extended in all counter calculations.	64 {x}

Access

MRS <Xt>, CNTVOFF_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0000	0b011

MSR CNTVOFF_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0000	0b011

Accessibility

MRS <Xt>, CNTVOFF_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = CNTVOFF_EL2;

```

MSR CNTVOFF_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTVOFF_EL2 = X[t, 64];

```

A.2.10.12 CNTHCTL_EL2, Counter-timer Hypervisor Control Register

Controls the generation of an event stream from the physical counter, and access from EL1 to the physical counter and the EL1 physical timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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

The following field descriptions apply in all Armv8.0 implementations.

Figure A-461: AARCH64_CNTHCTL_EL2 bit assignments

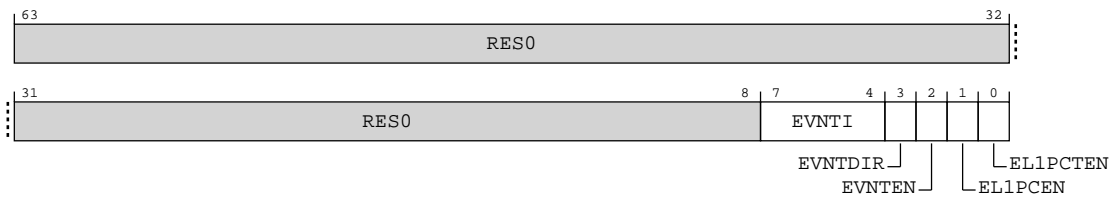


Table A-1190: CNTHCTL_EL2 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:4]	EVNTI	Selects which bit of AArch64-CNTPCT_ELO, as seen from EL2,is the trigger for the event stream generated from that counter when that stream is enabled. This field selects a trigger bit in the range 0 to 15 of AArch64-CNTPCT_ELO.	xxxx
[3]	EVNTDIR	Controls which transition of the AArch64-CNTPCT_ELO trigger bit, as seen from EL2 and defined by EVNTI, generates an event when the event stream is enabled. 0b0 A 0 to 1 transition of the trigger bit triggers an event. 0b1 A 1 to 0 transition of the trigger bit triggers an event.	x

Bits	Name	Description	Reset
[2]	EVNTEN	Enables the generation of an event stream from AArch64-CNTPCT_ELO as seen from EL2. 0b0 Disables the event stream. 0b1 Enables the event stream.	x
[1]	EL1PCEN	Traps ELO and EL1 accesses to the EL1 physical timer registers to EL2 when EL2 is enabled in the current Security state, as follows: <ul style="list-style-type: none"> In AArch64 state, accesses to AArch64-CNTP_CTL_ELO, AArch64-CNTP_CVAL_ELO, AArch64-CNTP_TVAL_ELO are trapped to EL2, reported using EC syndrome value 0x18. 0b0 From AArch64 state: ELO and EL1 accesses to the AArch64-CNTP_CTL_ELO, AArch64-CNTP_CVAL_ELO, and AArch64-CNTP_TVAL_ELO are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by AArch64-CNTKCTL_EL1.ELOPTEN. 0b1 This control does not cause any instructions to be trapped.	x
[0]	EL1PCTEN	Traps ELO and EL1 accesses to the EL1 physical counter register to EL2 when EL2 is enabled in the current Security state, as follows: <ul style="list-style-type: none"> In AArch64 state, accesses to AArch64-CNTPCT_ELO are trapped to EL2, reported using EC syndrome value 0x18. 0b0 From AArch64 state: ELO and EL1 accesses to the AArch64-CNTPCT_ELO are trapped to EL2 when EL2 is enabled in the current Security state, unless they are trapped by AArch64-CNTKCTL_EL1.ELOPCTEN. 0b1 This control does not cause any instructions to be trapped.	x

Access

MRS <Xt>, CNTHCTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b11110	0b0001	0b000

MSR CNTHCTL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b11110	0b0001	0b000

Accessibility

MRS <Xt>, CNTHCTL_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CNTHCTL_EL2;

```

MSR CNTHCTL_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHCTL_EL2 = X[t, 64];
```

A.2.10.13 CNTHPS_TVAL_EL2, Counter-timer Secure Physical Timer TimerValue Register (EL2)

Holds the timer value for the Secure EL2 physical timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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	0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-462: AARCH64_CNTHPS_TVAL_EL2 bit assignments

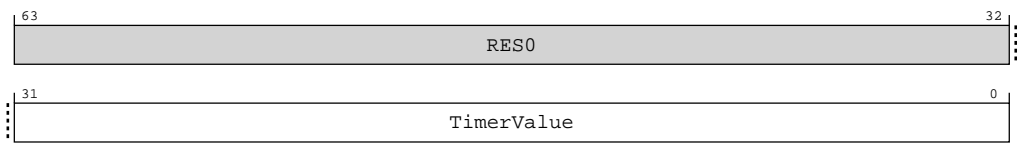


Table A-1193: CNTHPS_TVAL_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	TimerValue	<p>The TimerValue view of the EL2 physical timer.</p> <p>On a read of this register:</p> <ul style="list-style-type: none"> If AArch64-CNTHPS_CTL_EL2.ENABLE is 0, the value returned is UNKNOWN. If AArch64-CNTHPS_CTL_EL2.ENABLE is 1, the value returned is (AArch64-CNTHPS_CVAL_EL2 - AArch64-CNTPCT_ELO). <p>On a write of this register, AArch64-CNTHPS_CVAL_EL2 is set to (AArch64-CNTPCT_ELO + TimerValue), where TimerValue is treated as a signed 32-bit integer.</p> <p>When AArch64-CNTHPS_CTL_EL2.ENABLE is 1, the timer condition is met when (AArch64-CNTPCT_ELO - AArch64-CNTHPS_CVAL_EL2) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:</p> <ul style="list-style-type: none"> AArch64-CNTHPS_CTL_EL2.ISTATUS is set to 1. If AArch64-CNTHPS_CTL_EL2.IMASK is 0, an interrupt is generated. <p>When AArch64-CNTHPS_CTL_EL2.ENABLE is 0, the timer condition is not met, but AArch64-CNTPCT_ELO continues to count, so the TimerValue view appears to continue to count down.</p>	32 {x}

Access

MRS <Xt>, CNTHPS_TVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b11110	0b0101	0b000

MSR CNTHPS_TVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b11110	0b0101	0b000

Accessibility

MRS <Xt>, CNTHPS_TVAL_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    if CNTHPS_CTL_EL2.ENABLE == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = CNTHPS_CVAL_EL2 - PhysicalCountInt();

```

MSR CNTHPS_TVAL_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```
    UNDEFINED;
    elsif PSTATE.EL == EL2 then
        CNTHPS_CVAL_EL2 = SignExtend(X[t, 64]<31:0>, 64) + PhysicalCountInt();
```

A.2.10.14 CNTHPS_CTL_EL2, Counter-timer Secure Physical Timer Control Register (EL2)

Control register for the Secure EL2 physical timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

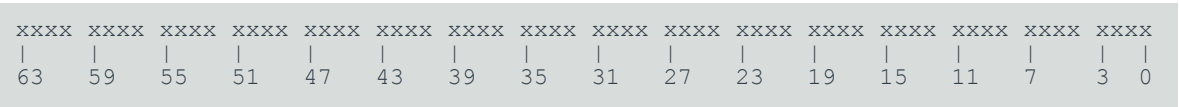
Functional group

Generic Timer registers

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-463: AARCH64_CNTHPS_CTL_EL2 bit assignments

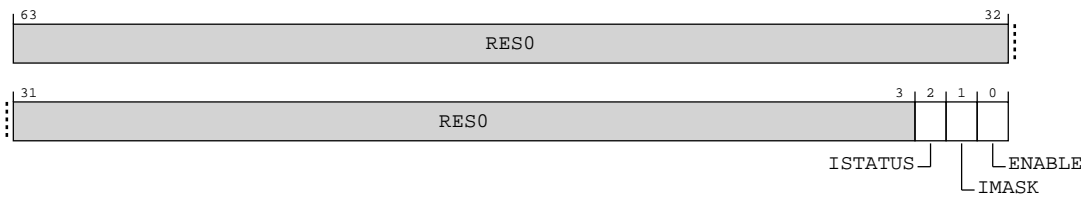


Table A-1196: CNTHPS_CTL_EL2 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2]	ISTATUS	<p>The status of the timer. This bit indicates whether the timer condition is met:</p> <p>0b0 Timer condition is not met.</p> <p>0b1 Timer condition is met.</p> <p>When the value of the CNTHPS_CTL_EL2.ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.</p> <p>When the value of the CNTHPS_CTL_EL2.ENABLE bit is 0, the ISTATUS field is UNKNOWN.</p> <p>Access to this field is: RO</p>	x
[1]	IMASK	<p>Timer interrupt mask bit. Permitted values are:</p> <p>0b0 Timer interrupt is not masked by the IMASK bit.</p> <p>0b1 Timer interrupt is masked by the IMASK bit.</p> <p>For more information, see the description of the ISTATUS bit.</p>	x
[0]	ENABLE	<p>Enables the timer. Permitted values are:</p> <p>0b0 Timer disabled.</p> <p>0b1 Timer enabled.</p> <p>Setting this bit to 0 disables the timer output signal, but the timer value accessible from AArch64-CNTHPS_TVAL_EL2 continues to count down.</p> <p>Note: Disabling the output signal might be a power-saving option.</p>	x

Access

MRS <Xt>, CNTHPS_CTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0101	0b001

MSR CNTHPS_CTL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0101	0b001

Accessibility

MRS <Xt>, CNTHPS_CTL_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CNTHPS_CTL_EL2;
```

MSR CNTHPS_CTL_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    CNTHPS_CTL_EL2 = X[t, 64];
```

A.2.10.15 CNTHPS_CVAL_EL2, Counter-timer Secure Physical Timer CompareValue Register (EL2)

Holds the compare value for the Secure EL2 physical timer.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic Timer 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-464: AARCH64_CNTHPS_CVAL_EL2 bit assignments

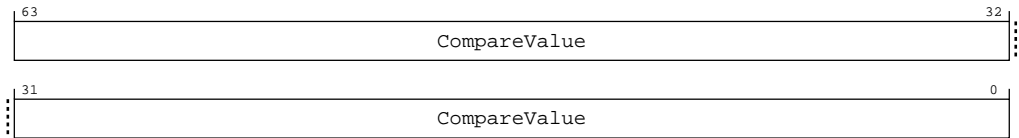


Table A-1199: CNTHPS_CVAL_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	CompareValue	<p>Holds the EL2 physical timer CompareValue.</p> <p>When AArch64-CNTHPS_CTL_EL2.ENABLE is 1, the timer condition is met when (AArch64-CNTPCT_ELO - CompareValue) is greater than or equal to zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met:</p> <ul style="list-style-type: none">AArch64-CNTHPS_CTL_EL2.ISTATUS is set to 1.If AArch64-CNTHPS_CTL_EL2.IMASK is 0, an interrupt is generated. <p>When AArch64-CNTHPS_CTL_EL2.ENABLE is 0, the timer condition is not met, but AArch64-CNTPCT_ELO continues to count.</p> <p>If the Generic counter is implemented at a size less than 64 bits, then this field is permitted to be implemented at the same width as the counter, and the upper bits are RES0.</p> <p>The value of this field is treated as zero-extended in all counter calculations.</p>	64 {x}

Access

MRS <Xt>, CNTHPS_CVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0101	0b010

MSR CNTHPS_CVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0101	0b010

Accessibility

MRS <Xt>, CNTHPS_CVAL_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = CNTHPS_CVAL_EL2;
```

MSR CNTHPS_CVAL_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHPS_CVAL_EL2 = X[t, 64];
```

A.2.11 AArch64 other register description

This section includes the register descriptions for all other registers in the Cortex®-R82 processor.

A.2.11.1 SCTLR_EL2, System Control Register (EL2)

Provides top level control of the system, including its memory system, at EL2.

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	xx0x	xxx0	xxxx	xxxx	x0x0
63	59	55	51	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-465: AARCH64_SCTLR_EL2 bit assignments

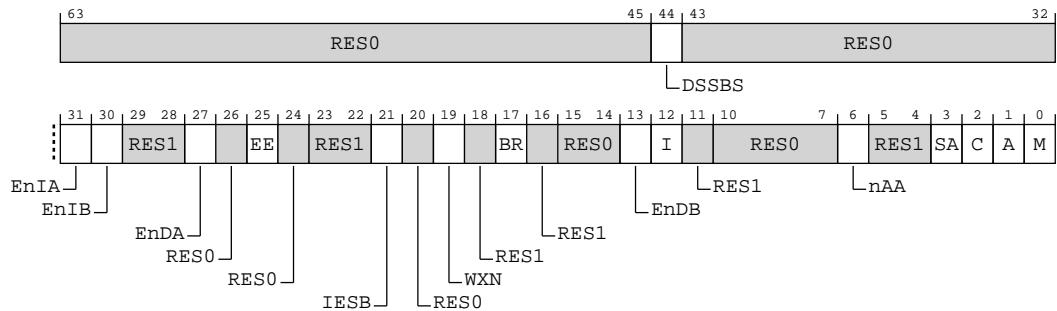


Table A-1202: SCTLR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:45]	RES0	Reserved	RES0
[44]	DSSBS	Default PSTATE.SSBS value on Exception Entry. 0b0 PSTATE.SSBS is set to 0 on an exception to EL2. 0b1 PSTATE.SSBS is set to 1 on an exception to EL2.	x
[43:32]	RES0	Reserved	RES0
[31]	EnIA	Controls enabling of pointer authentication of instruction addresses, using the APIAKey_EL1 key, in the EL2 or EL2&0 translation regime. 0b0 Pointer authentication of instruction addresses, using the APIAKey_EL1 key, is not enabled. 0b1 Pointer authentication of instruction addresses, using the APIAKey_EL1 key, is enabled. Note: This field controls the behavior of the AddPACIA and AuthIA pseudocode functions. Specifically, when the field is 1, AddPACIA returns a copy of a pointer to which a pointer authentication code has been added, and AuthIA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP .	x
[30]	EnIB	Controls enabling of pointer authentication of instruction addresses, using the APIBKey_EL1 key, in the EL2 or EL2&0 translation regime. 0b0 Pointer authentication of instruction addresses, using the APIBKey_EL1 key, is not enabled. 0b1 Pointer authentication of instruction addresses, using the APIBKey_EL1 key, is enabled. Note: This field controls the behavior of the AddPACIB and AuthIB pseudocode functions. Specifically, when the field is 1, AddPACIB returns a copy of a pointer to which a pointer authentication code has been added, and AuthIB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP .	x

Bits	Name	Description	Reset
[29:28]	RES1	Reserved	RES1
[27]	EnDA	Controls enabling of pointer authentication of instruction addresses, using the APDAKey_EL1 key, in the EL2 or EL2&0 translation regime. 0b0 Pointer authentication of data addresses, using the APDAKey_EL1 key, is not enabled. 0b1 Pointer authentication of data addresses, using the APDAKey_EL1 key, is enabled. Note: This field controls the behavior of the AddPACDA and AuthDA pseudocode functions. Specifically, when the field is 1, AddPACDA returns a copy of a pointer to which a pointer authentication code has been added, and AuthDA returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP .	x
[26]	RES0	Reserved	RES0
[25]	EE	Endianness of data accesses at EL2. 0b0 Explicit data accesses at EL2 are little-endian. 0b1 Explicit data accesses at EL2 are big-endian. The EE bit is permitted to be cached in a TLB.	x
[24]	RES0	Reserved	RES0
[23:22]	RES1	Reserved	RES1
[21]	IESB	Implicit Error Synchronization event enable. 0b0 Disabled. 0b1 An implicit error synchronization event is added: <ul style="list-style-type: none"> At each exception taken to EL2. Before the operational pseudocode of each ERET instruction executed at EL2. When the PE is in Debug state, this field has no effect.	x
[20]	RES0	Reserved	RES0
[19]	WXN	Write permission implies XN (Execute-never). For the EL2 or EL2&0 translation regime, this bit can force all memory regions that are writable to be treated as XN. 0b0 This control has no effect on memory access permissions. 0b1 Any region that is writable in the EL2 or EL2&0 translation regime is forced to XN for accesses from software executing at EL2. This bit applies only when SCTLR_EL2.M bit is set. The WXN bit is permitted to be cached in a TLB.	x
[18]	RES1	Reserved	RES1

Bits	Name	Description	Reset
[17]	BR	Background region enable for EL2 MPU memory regions. 0b0 Background region disabled for stage 1 EL2 translation regime and stage 2 EL1&0 translation regime. 0b1 Background region enabled for stage 1 EL2 translation regime and stage 2 EL1&0 translation regime. If EL2 MPU is enabled, then EL0 and EL1 access that does not match an EL2 MPU region always results in a Translation fault.	0b0
[16]	RES1	Reserved	RES1
[15:14]	RES0	Reserved	RES0
[13]	EnDB	Controls enabling of pointer authentication of instruction addresses, using the APDBKey_EL1 key, in the EL2 or EL2&0 translation regime. 0b0 Pointer authentication of data addresses, using the APDBKey_EL1 key, is not enabled. 0b1 Pointer authentication of data addresses, using the APDBKey_EL1 key, is enabled. Note: This field controls the behavior of the AddPACDB and AuthDB pseudocode functions. Specifically, when the field is 1, AddPACDB returns a copy of a pointer to which a pointer authentication code has been added, and AuthDB returns an authenticated copy of a pointer. When the field is 0, both of these functions are NOP .	x
[12]	I	Instruction access Cacheability control, for accesses at EL2. 0b0 All instruction accesses to Normal memory from EL2 are Non-cacheable for all levels of instruction and unified cache. If SCTL_EL2.{BR, M} == {0, 0}, then instruction accesses from stage 1 of the EL2 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory. 0b1 This control has no effect on the Cacheability of instruction access to Normal memory from EL2. If SCTL_EL2.{BR, M} = {0, 0}, then instruction accesses from stage 1 of the EL2 translation regime are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory. This bit has no effect on the EL1&0 translation regime.	0b0
[11]	RES1	Reserved	RES1
[10:7]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[6]	nAA	<p>Non-aligned access. This bit controls generation of Alignment faults at EL2 under certain conditions.</p> <p>The following instructions generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for access:</p> <ul style="list-style-type: none"> LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH. STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH <p>0b0 Unaligned accesses by the specified instructions generate an Alignment fault.</p> <p>0b1 Unaligned accesses by the specified instructions do not generate an Alignment fault.</p>	x
[5:4]	RES1	Reserved	RES1
[3]	SA	SP Alignment check enable. When set to 1, if a load or store instruction executed at EL2 uses the SP as the base address and the SP is not aligned to a 16-byte boundary, then an SP alignment fault exception is generated. For more information, see <i>SP alignment checking</i> in the Arm® Architecture Reference Manual for A-profile architecture .	x
[2]	C	<p>Data access Cacheability control, for accesses at EL2.</p> <p>0b0 All data accesses to Normal memory from EL2 are Non-cacheable for all levels of data and unified cache.</p> <p>0b1 This control has no effect on the Cacheability of data accesses to Normal memory from EL2.</p> <p>This bit has no effect on the EL1&0 translation regime.</p>	0b0
[1]	A	<p>Alignment check enable. This is the enable bit for Alignment fault checking at EL2.</p> <p>0b0 Alignment fault checking is disabled when executing at EL2.</p> <p>Alignment checks on some instructions are not disabled by this control. For more information, see <i>Alignment of data accesses</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>0b1 Alignment fault checking is enabled when executing at EL2.</p> <p>All instructions that load or store one or more registers have an alignment check that the address being accessed is aligned to the size of the data element(s) being accessed. If this check fails it causes an Alignment fault, which is taken as a Data Abort exception.</p>	x
[0]	M	<p>MPU enable for EL2 stage 1 and EL1&0 stage 2 address translation.</p> <p>0b0 MPU disabled for EL2 and EL1&0 stage 2 address translation.</p> <p>See the SCTLR_EL2.I field for the behavior of instruction accesses to Normal memory.</p> <p>0b1 MPU enabled for EL2 and EL1&0 stage 2 address translation.</p>	0b0

Access

MRS <Xt>, SCTLR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b000

MSR SCTLR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b000

Accessibility

MRS <Xt>, SCTLR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = SCTLR_EL2;
```

MSR SCTLR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    SCTLR_EL2 = X[t, 64];
```

A.2.11.2 HSTR_EL2, Hypervisor System Trap Register

Controls trapping to EL2 of EL1 or lower AArch32 accesses to the System register in the coproc == 0b1111 encoding space, by the CRn value used to access the register using MCR or MRC instruction. When the register is accessible using an MCRR or MRRC instruction, this is the CRm value used to access the register.

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
```




Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure A-466: AARCH64_HSTR_EL2 bit assignments

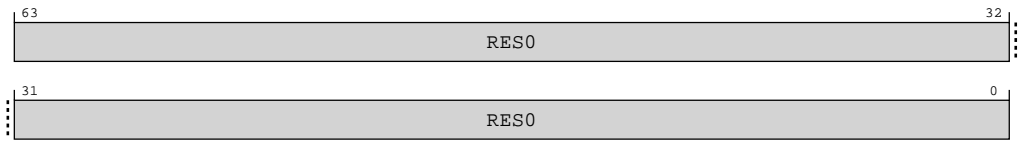


Table A-1205: HSTR_EL2 bit descriptions

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

Access

MRS <Xt>, HSTR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b011

MSR HSTR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0001	0b011

Accessibility

MRS <Xt>, HSTR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    X[t, 64] = HSTR_EL2;
```

MSR HSTR_EL2, <Xt>

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

```
elseif PSTATE.EL == EL2 then  
    HSTR_EL2 = X[t, 64];
```

Appendix B External registers

This appendix contains the descriptions for all the external (memory-mapped) registers in the Cortex®-R82 processor.

B.1 Registers accessed over the Utility bus

This section contains the summary and descriptions for all the external registers in the Cortex®-R82 processor accessed over the Utility bus.

Utility bus implements one of two memory maps:

- A sparse memory map, when the configuration parameter `DENSE_CS_ADDR_MAP` is 0. Each set of System registers is grouped on separate 64KB page boundaries. This allows to isolate and remap components when using the EL1 MMU (if VMSA included) with the maximum 64KB granule.
- A dense memory map, when the configuration parameter `DENSE_CS_ADDR_MAP` is 1. Each set of System registers is grouped on separate 4KB page boundaries. It may not be possible to isolate and remap individual components using an EL1 MMU with 16KB or 64KB granules. You can either use a 4KB granule, or depend on an EL2 MPU (or an EL1 MPU, if EL1 is using PMSA) to isolate components without remapping.

The following table shows the total space occupied by the Utility bus memory map, depending on how many cores are implemented in the processor.

Table B-1: Utility bus sparse and dense memory map sizes

Number of cores	Sparse memory map	Dense memory map
1	20-bit (1MB)	15-bit (32KB)
2	21-bit (2MB)	15-bit (32KB)
3	22-bit (4MB)	16-bit (64KB)
4	22-bit (4MB)	16-bit (64KB)
5	23-bit (8MB)	17-bit (128KB)
6	23-bit (8MB)	17-bit (128KB)
7	23-bit (8MB)	17-bit (128KB)
8	23-bit (8MB)	17-bit (128KB)

The following table shows the base addresses for each set of system component registers that the external agents can access using the Utility bus.



Note

- In the following table, any address space that is not documented or that corresponds to non-implemented cores 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 B-2: Memory map for external agents accessing the Utility bus

Base address, sparse map (DENSE_CS_ADDR_MAP = 0)	Base address, dense map (DENSE_CS_ADDR_MAP = 1)	Registers	Memory map
0x010000	0x01000	Cluster PPU	B.1.1.3 External CLUSTERPPU registers summary on page 1419
0x020000	0x02000	Core 0 RAS	B.1.1.1 External RAS registers summary on page 1416
0x040000	0x04000	Core 0 PPU	B.1.1.2 External PPU registers summary on page 1417
0x120000	0x05000	Core 1 RAS	B.1.1.1 External RAS registers summary on page 1416
0x140000	0x07000	Core 1 PPU	B.1.1.2 External PPU registers summary on page 1417
0x220000	0x08000	Core 2 RAS	B.1.1.1 External RAS registers summary on page 1416
0x240000	0x0A000	Core 2 PPU	B.1.1.2 External PPU registers summary on page 1417
0x320000	0x0B000	Core 3 RAS	B.1.1.1 External RAS registers summary on page 1416
0x340000	0x0D000	Core 3 PPU	B.1.1.2 External PPU registers summary on page 1417
0x420000	0x0E000	Core 4 RAS	B.1.1.1 External RAS registers summary on page 1416
0x440000	0x10000	Core 4 PPU	B.1.1.2 External PPU registers summary on page 1417
0x520000	0x11000	Core 5 RAS	B.1.1.1 External RAS registers summary on page 1416
0x540000	0x13000	Core 5 PPU	B.1.1.2 External PPU registers summary on page 1417
0x620000	0x14000	Core 6 RAS	B.1.1.1 External RAS registers summary on page 1416
0x640000	0x16000	Core 6 PPU	B.1.1.2 External PPU registers summary on page 1417
0x720000	0x17000	Core 7 RAS	B.1.1.1 External RAS registers summary on page 1416
0x740000	0x19000	Core 7 PPU	B.1.1.2 External PPU registers summary on page 1417

B.1.1 Register summaries for registers accessed over the Utility bus

This section includes the summary tables for all the external registers in the Cortex®-R82 processor accessed over the Utility bus.

B.1.1.1 External RAS registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-3: RAS registers summary

Offset	Name	Reset	Width	Description
0x000 + (64 * n)	ERR<n>FR	See individual bit resets.	64-bit	Error Record <n> Feature Register
0x008 + (64 * n)	ERR<n>CTLR	See individual bit resets.	64-bit	Error Record <n> Control Register
0x010 + (64 * n)	ERR<n>STATUS	See individual bit resets.	64-bit	Error Record <n> Primary Status Register
0x020 + (64 * n)	ERR<n>MISC0	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 0
0x028 + (64 * n)	ERR<n>MISC1	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 1
0x030 + (64 * n)	ERR<n>MISC2	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 2
0x038 + (64 * n)	ERR<n>MISC3	See individual bit resets.	64-bit	Error Record <n> Miscellaneous Register 3
0x800 + (64 * n)	ERR<n>PFGF	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Feature Register
0x808 + (64 * n)	ERR<n>PFGCTL	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Control Register
0x810 + (64 * n)	ERR<n>PFGCDN	See individual bit resets.	64-bit	Error Record <n> Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	See individual bit resets.	64-bit	Error Group Status Register
0xE10	ERRIIDR	See individual bit resets.	32-bit	Implementation Identification Register
0xFA8	ERRDEVAFF	See individual bit resets.	64-bit	Device Affinity Register
0xFBC	ERRDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC8	ERRDEVID	See individual bit resets.	32-bit	Device Configuration Register
0xFD0	ERRPIDR4	See individual bit resets.	32-bit	Peripheral Identification Register 4
0xFE0	ERRPIDR0	See individual bit resets.	32-bit	Peripheral Identification Register 0
0xFE4	ERRPIDR1	See individual bit resets.	32-bit	Peripheral Identification Register 1
0xFE8	ERRPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	ERRPIDR3	See individual bit resets.	32-bit	Peripheral Identification Register 3
0xFF0	ERRCIDR0	See individual bit resets.	32-bit	Component Identification Register 0
0xFF4	ERRCIDR1	See individual bit resets.	32-bit	Component Identification Register 1
0xFF8	ERRCIDR2	See individual bit resets.	32-bit	Component Identification Register 2
0xFFC	ERRCIDR3	See individual bit resets.	32-bit	Component Identification Register 3

B.1.1.2 External PPU registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped PPU 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-4: PPU registers summary

Offset	Name	Reset	Width	Description
0x000	PPU_PWPR	See individual bit resets.	32-bit	Power Policy Register
0x004	PPU_PMER	See individual bit resets.	32-bit	Power Mode Emulation Enable Register
0x008	PPU_PWSR	See individual bit resets.	32-bit	Power Status Register
0x010	PPU_DISR	See individual bit resets.	32-bit	Device Interface Input Current Status Register
0x014	PPU_MISR	See individual bit resets.	32-bit	Miscellaneous Input Current Status Register
0x018	PPU_STSR	See individual bit resets.	32-bit	Stored Status Register
0x01C	PPU_UNLK	See individual bit resets.	32-bit	Unlock Register
0x020	PPU_PWCR	See individual bit resets.	32-bit	Power Configuration Register
0x024	PPU_PTCR	See individual bit resets.	32-bit	Power Mode Transition Register
0x030	PPU_IMR	See individual bit resets.	32-bit	Interrupt Mask Register
0x034	PPU_AIMR	See individual bit resets.	32-bit	Additional Interrupt Mask Register
0x038	PPU_ISR	See individual bit resets.	32-bit	Interrupt Status Register
0x03C	PPU_AISR	See individual bit resets.	32-bit	Additional Interrupt Status Register
0x040	PPU_IESR	See individual bit resets.	32-bit	Input Edge Sensitivity Register
0x044	PPU_OPSR	See individual bit resets.	32-bit	Operating Mode Active Edge Sensitivity Register
0x050	PPU_FUNRR	See individual bit resets.	32-bit	Functional Retention RAM Configuration Register
0x054	PPU_FULRR	See individual bit resets.	32-bit	Full Retention RAM Configuration Register
0x058	PPU_MEMRR	See individual bit resets.	32-bit	Memory Retention RAM Configuration Register
0x160	PPU_EDTR0	See individual bit resets.	32-bit	Power Mode Entry Delay Register 0
0x164	PPU_EDTR1	See individual bit resets.	32-bit	Power Mode Entry Delay Register 1
0x170	PPU_DCDR0	See individual bit resets.	32-bit	Device Control Delay Configuration Register 0
0x174	PPU_DCDR1	See individual bit resets.	32-bit	Device Control Delay Configuration Register 1
0xFB0	PPU_IDR0	See individual bit resets.	32-bit	PPU Identification Register 0
0xFB4	PPU_IDR1	See individual bit resets.	32-bit	PPU Identification Register 1
0xFC8	PPU_IIDR	See individual bit resets.	32-bit	Implementation Identification Register
0xFCC	PPU_AIDR	See individual bit resets.	32-bit	Architecture Identification Register
0xFD0	PPU_PIDR4	See individual bit resets.	32-bit	PPU Peripheral Identification Register 4
0xFD4	PPU_PIDR5	See individual bit resets.	32-bit	PPU Peripheral Identification Register 5
0xFD8	PPU_PIDR6	See individual bit resets.	32-bit	PPU Peripheral Identification Register 6
0xFDC	PPU_PIDR7	See individual bit resets.	32-bit	PPU Peripheral Identification Register 7

Offset	Name	Reset	Width	Description
0xFE0	PPU_PIDR0	See individual bit resets.	32-bit	PPU Peripheral Identification Register 0
0xFE4	PPU_PIDR1	See individual bit resets.	32-bit	PPU Peripheral Identification Register 1
0xFE8	PPU_PIDR2	See individual bit resets.	32-bit	PPU Peripheral Identification Register 2
0xFEC	PPU_PIDR3	See individual bit resets.	32-bit	PPU Peripheral Identification Register 3
0xFF0	PPU_CIDR0	See individual bit resets.	32-bit	PPU Component Identification Register 0
0xFF4	PPU_CIDR1	See individual bit resets.	32-bit	PPU Component Identification Register 1
0xFF8	PPU_CIDR2	See individual bit resets.	32-bit	PPU Component Identification Register 2
0xFFC	PPU_CIDR3	See individual bit resets.	32-bit	PPU Component Identification Register 3

B.1.1.3 External CLUSTERPPU registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped CLUSTERPPU 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-5: CLUSTERPPU registers summary

Offset	Name	Reset	Width	Description
0x000	CLUSTERPPU_PWPR	See individual bit resets.	32-bit	Power Policy Register
0x004	CLUSTERPPU_PMER	See individual bit resets.	32-bit	Power Mode Emulation Enable Register
0x008	CLUSTERPPU_PWSR	See individual bit resets.	32-bit	Power Status Register
0x010	CLUSTERPPU_DISR	See individual bit resets.	32-bit	Device Interface Input Current Status Register
0x014	CLUSTERPPU_MISR	See individual bit resets.	32-bit	Miscellaneous Input Current Status Register
0x018	CLUSTERPPU_STSR	See individual bit resets.	32-bit	Stored Status Register
0x01C	CLUSTERPPU_UNLK	See individual bit resets.	32-bit	Unlock Register
0x020	CLUSTERPPU_PWCR	See individual bit resets.	32-bit	Power Configuration Register
0x024	CLUSTERPPU_PTCR	See individual bit resets.	32-bit	Power Mode Transition Register
0x030	CLUSTERPPU_IMR	See individual bit resets.	32-bit	Interrupt Mask Register
0x034	CLUSTERPPU_AIMR	See individual bit resets.	32-bit	Additional Interrupt Mask Register
0x038	CLUSTERPPU_ISR	See individual bit resets.	32-bit	Interrupt Status Register
0x03C	CLUSTERPPU_AISR	See individual bit resets.	32-bit	Additional Interrupt Status Register
0x040	CLUSTERPPU_IESR	See individual bit resets.	32-bit	Input Edge Sensitivity Register
0x044	CLUSTERPPU_OPSR	See individual bit resets.	32-bit	Operating Mode Active Edge Sensitivity Register
0x050	CLUSTERPPU_FUNRR	See individual bit resets.	32-bit	Functional Retention RAM Configuration Register
0x054	CLUSTERPPU_FULRR	See individual bit resets.	32-bit	Full Retention RAM Configuration Register
0x058	CLUSTERPPU_MEMRR	See individual bit resets.	32-bit	Memory Retention RAM Configuration Register
0x160	CLUSTERPPU_EDTRO	See individual bit resets.	32-bit	Power Mode Entry Delay Register 0

Offset	Name	Reset	Width	Description
0x164	CLUSTERPPU_EDTR1	See individual bit resets.	32-bit	Power Mode Entry Delay Register 1
0x170	CLUSTERPPU_DCDR0	See individual bit resets.	32-bit	Device Control Delay Configuration Register 0
0x174	CLUSTERPPU_DCDR1	See individual bit resets.	32-bit	Device Control Delay Configuration Register 1
0xFB0	CLUSTERPPU_IDR0	See individual bit resets.	32-bit	PPU Identification Register 0
0xFB4	CLUSTERPPU_IDR1	See individual bit resets.	32-bit	PPU Identification Register 1
0xFC8	CLUSTERPPU_IIDR	See individual bit resets.	32-bit	Implementation Identification Register
0xFCC	CLUSTERPPU_AIDR	See individual bit resets.	32-bit	Architecture Identification Register
0xFD0	CLUSTERPPU_PIDR4	See individual bit resets.	32-bit	PPU Peripheral Identification Register 4
0xFD4	CLUSTERPPU_PIDR5	See individual bit resets.	32-bit	PPU Peripheral Identification Register 5
0xFD8	CLUSTERPPU_PIDR6	See individual bit resets.	32-bit	PPU Peripheral Identification Register 6
0xFDC	CLUSTERPPU_PIDR7	See individual bit resets.	32-bit	PPU Peripheral Identification Register 7
0xFE0	CLUSTERPPU_PIDR0	See individual bit resets.	32-bit	PPU Peripheral Identification Register 0
0xFE4	CLUSTERPPU_PIDR1	See individual bit resets.	32-bit	PPU Peripheral Identification Register 1
0xFE8	CLUSTERPPU_PIDR2	See individual bit resets.	32-bit	PPU Peripheral Identification Register 2
0xFEC	CLUSTERPPU_PIDR3	See individual bit resets.	32-bit	PPU Peripheral Identification Register 3
0xFF0	CLUSTERPPU_CIDR0	See individual bit resets.	32-bit	CLUSTERPPU Component Identification Register 0
0xFF4	CLUSTERPPU_CIDR1	See individual bit resets.	32-bit	CLUSTERPPU Component Identification Register 1
0xFF8	CLUSTERPPU_CIDR2	See individual bit resets.	32-bit	CLUSTERPPU Component Identification Register 2
0xFFC	CLUSTERPPU_CIDR3	See individual bit resets.	32-bit	CLUSTERPPU Component Identification Register 3

B.1.2 Register descriptions for registers accessed over the Utility bus

This section includes the descriptions for all the external registers in the Cortex®-R82 processor accessed over the Utility bus.

B.1.2.1 External RAS register description

This section includes the register descriptions for all memory-mapped *Reliability, Availability, and Serviceability* (RAS) registers that are accessed for each core in the Cortex®-R82 processor.

B.1.2.1.1 ERR<n>FR, Error Record <n> Feature Register, n = 0 - 9

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

Present only if error record <n> is implemented. Otherwise, this register is RES0.

1 or 7 error records are implemented, depending on the RAM_PROTECTION parameter.

Attributes

Width

64

Component

RAS

Register offset

0x000 + (64 * n)

Access type

RO

Reset value

When n == 0, or (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00 10xx xx10 xxxx 1001 1010 xx10
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx00



Where the reset reads xxxx, see individual bits.

Bit descriptions

When n == 0, or (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

Figure B-1: EXT_ERR<n>FR bit assignments

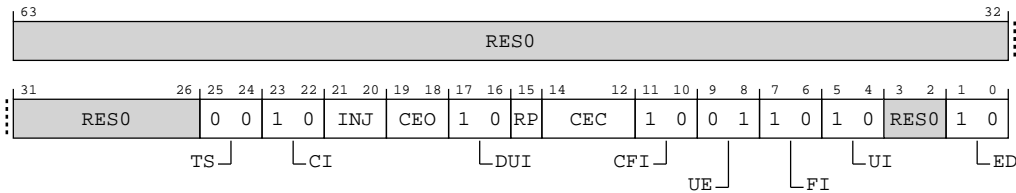


Table B-6: ERR<n>FR bit descriptions

Bits	Name	Description	Reset
[63:26]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[25:24]	TS	<p>Timestamp Extension.</p> <p>Indicates whether, for each error record <m> owned by this node, ERR<m>MISC3 is used as the timestamp register, and, if it is, the timebase used by the timestamp.</p> <p>0b00</p> <p>The node does not support a timestamp register.</p> <p>All other values are reserved.</p>	0b00
[23:22]	CI	<p>Critical error interrupt.</p> <p>Indicates whether the critical error interrupt and associated controls are implemented.</p> <p>0b10</p> <p>Critical error interrupt is supported and it can be enabled using associated controls.</p> <p>All other values are reserved.</p>	0b10
[21:20]	INJ	<p>When n == 1 or n == 4</p> <p>Fault Injection Extension.</p> <p>Indicates whether the RAS Common Fault Injection Model Extension is implemented.</p> <p>01</p> <p>The node implements the RAS Common Fault Injection Model Extension. See ext-ERR<n>PFGF for more information.</p> <p>When n == 0</p> <p>Fault Injection Extension.</p> <p>Indicates whether the RAS Common Fault Injection Model Extension is implemented.</p> <p>00</p> <p>The node does not support the RAS Common Fault Injection Model Extension.</p> <p>Otherwise</p> <p>RES0</p>	xx
[19:18]	CEO	<p>When n == 1 or n == 4</p> <p>Corrected Error overwrite. Indicates the behavior of the node when a second or subsequent Corrected error is recorded and a first Corrected error has previously been recorded by an error record <m> owned by the node.</p> <p>00</p> <p>Keeps the previous error syndrome.</p> <p>Otherwise</p> <p>RES0</p>	xx
[17:16]	DUI	<p>Error recovery interrupt for deferred errors.</p> <p>Indicates whether the node implements a control for enabling error recovery interrupts on deferred errors.</p> <p>0b10</p> <p>Feature is controllable using ext-ERR<n>CTLR.DUI.</p>	0b10

Bits	Name	Description	Reset
[15]	RP	<p>When n == 1 or n == 4</p> <p>Repeat counter. Indicates whether the node implements a second Corrected error counter in ERR<m>MISCO for each error record <m> owned by the node that can record countable errors.</p> <p>0</p> <p>Implements a single Corrected error counter in ERR<m>MISCO for each error record <m> owned by the node that can record countable errors.</p> <p>Otherwise</p> <p>RES0</p>	x
[14:12]	CEC	<p>When n == 1 or n == 4</p> <p>Corrected Error Counter.</p> <p>Indicates whether the node implements standard Corrected error counter (CE counter) mechanisms in ERR<m>MISCO for each error record <m> owned by the node that can record countable errors.</p> <p>010</p> <p>Implements an 8-bit Corrected error counter in ERR<m>MISCO[39:32].</p> <p>When n == 0</p> <p>Corrected Error Counter.</p> <p>Indicates whether the node implements standard Corrected error counter (CE counter) mechanisms in ERR<m>MISCO for each error record <m> owned by the node that can record countable errors.</p> <p>000</p> <p>Does not implement the standard Corrected error counter model.</p> <p>Otherwise</p> <p>RES0</p>	xxx
[11:10]	CFI	<p>Fault handling interrupt for corrected errors.</p> <p>Indicates whether the node implements a control for enabling fault handling interrupts on corrected errors.</p> <p>0b10</p> <p>Feature is controllable using ext-ERR<n>CTLR.CFI.</p>	0b10
[9:8]	UE	<p>In-band uncorrected error reporting.</p> <p>Indicates whether the node implements in-band uncorrected error reporting (External aborts), and, if so, whether the node implements controls for enabling and disabling the reporting.</p> <p>0b01</p> <p>Feature always enabled. ext-ERR<n>CTLR.UE is RES0.</p>	0b01
[7:6]	FI	<p>Fault handling interrupt.</p> <p>Indicates whether the node implements a fault handling interrupt, and, if so, whether the node implements controls for enabling and disabling the interrupt.</p> <p>0b10</p> <p>Feature is controllable using ext-ERR<n>CTLR.FI.</p>	0b10
[5:4]	UI	<p>Error recovery interrupt for uncorrected errors.</p> <p>Indicates whether the node implements an error recovery interrupt, and, if so, whether the node implements controls for enabling and disabling the interrupt.</p> <p>0b10</p> <p>Feature is controllable using ext-ERR<n>CTLR.UI.</p>	0b10

Bits	Name	Description	Reset
[3:2]	RES0	Reserved	RES0
[1:0]	ED	Error reporting and logging. Indicates whether <n> is the first record owned by the node, and, if it is, whether the node implements controls for enabling and disabling error reporting and logging. 0b10 Feature is controllable using ext-ERR<n>CTLR.ED.	0b10

Figure B-2: EXT_ERR<n>FR bit assignments

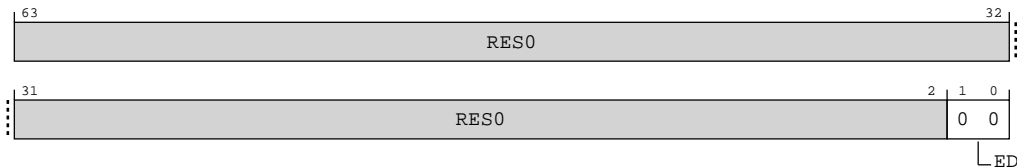


Table B-7: ERR<n>FR bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	RES0
[1:0]	ED	Error reporting and logging. Indicates whether <n> is the first record owned by the node, and, if it is, whether the node implements controls for enabling and disabling error reporting and logging. 0b00 Error record <n> is not the first record owned by the node.	0b00

Accessibility

Component	Offset	Instance	Range
RAS	0x000 + (64 * n)	ERR<n>FR	None

This interface is accessible as follows:

RO

B.1.2.1.2 ERR<n>CTLR, Error Record <n> Control Register, n = 0 - 9

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

1 or 7 error records are implemented, depending on the RAM_PROTECTION parameter.

If error record <n> is not implemented, or error record <n> is not the first error record owned by the node, ERR<n>CTLR is RES0.

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x008 + (64 * n)

Access type

RW

Reset value

When n == 0, or (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xx0x x0x0 xxxx 00x0
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 n == 0, or (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

Figure B-3: EXT_ERR<n>CTLR bit assignments

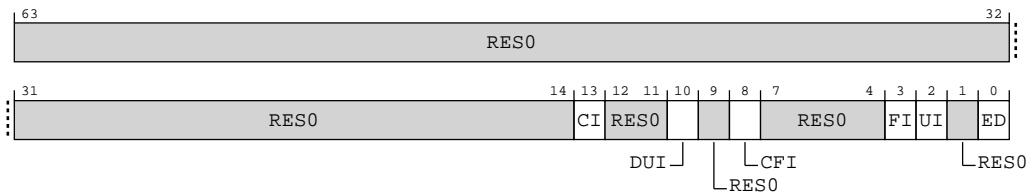


Table B-9: ERR<n>CTLR bit descriptions

Bits	Name	Description	Reset
[63:14]	RES0	Reserved	RES0
[13]	CI	<p>Critical error interrupt enable.</p> <p>When enabled, the critical error interrupt is generated for a critical error condition.</p> <p>0b0</p> <p>Critical error interrupt not generated for critical errors. Critical errors are treated as Uncontained errors.</p> <p>0b1</p> <p>Critical error interrupt generated for critical errors.</p>	0b0
[12:11]	RES0	Reserved	RES0
[10]	DUI	<p>Error recovery interrupt for deferred errors enable. This control applies to errors arising from both reads and writes.</p> <p>When enabled, an error recovery interrupt is generated for all detected Deferred errors.</p> <p>0b0</p> <p>Error recovery interrupt not generated for deferred errors.</p> <p>0b1</p> <p>Error recovery interrupt generated for deferred errors.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p>	0b0
[9]	RES0	Reserved	RES0
[8]	CFI	<p>Fault handling interrupt for Corrected errors enable. This control applies to errors arising from both reads and writes.</p> <p>When enabled:</p> <ul style="list-style-type: none"> If the node implements Corrected error counters, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 1. For more information, see ext-ERR<n>MISCO. Otherwise, the fault handling interrupt is generated for all detected Corrected errors. <p>0b0</p> <p>Fault handling interrupt not generated for Corrected errors.</p> <p>0b1</p> <p>Fault handling interrupt generated for Corrected errors.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p>	0b0
[7:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3]	FI	<p>Fault handling interrupt enable. This control applies to errors arising from both reads and writes.</p> <p>When enabled:</p> <ul style="list-style-type: none"> The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. If the fault handling interrupt for Corrected errors control is not implemented: <ul style="list-style-type: none"> If the node implements Corrected error counters, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 1. Otherwise, the fault handling interrupt is also generated for all detected Corrected errors. <p>0b0 Fault handling interrupt disabled.</p> <p>0b1 Fault handling interrupt enabled.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p>	0b0
[2]	UI	<p>Uncorrected error recovery interrupt enable. This control applies to errors arising from both reads and writes.</p> <p>When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred.</p> <p>0b0 Error recovery interrupt disabled.</p> <p>0b1 Error recovery interrupt enabled.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p>	0b0
[1]	RES0	Reserved	RES0
[0]	ED	<p>Error reporting and logging enable.</p> <p>When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. Correct error detection and correction codes are written for writes, unless error injection is taking place.</p> <p>0b0 Error reporting disabled.</p> <p>0b1 Error reporting enabled.</p> <p>Even when reporting and logging are disabled, the processor still takes all appropriate actions to correct, defer or abort on detected errors.</p>	0b0

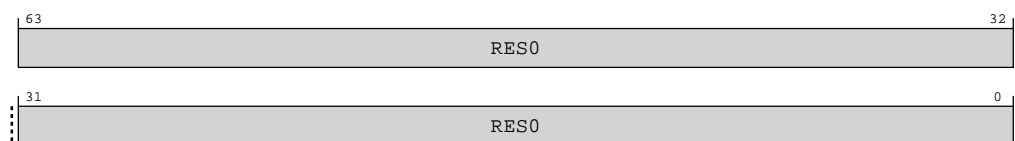
Figure B-4: EXT_ERR<n>CTLR bit assignments

Table B-10: ERR<n>CTLR bit descriptions

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

Accessibility

Component	Offset	Instance	Range
RAS	0x008 + (64 * n)	ERR<n>CTLR	None

This interface is accessible as follows:

RW

B.1.2.1.3 ERR<n>STATUS, Error Record <n> Primary Status Register, n = 0 - 9

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

1 or 7 error records are implemented, depending on the RAM_PROTECTION parameter.

If error record <n> is not implemented, ERR<n>STATUS is RES0.

ERR<q>FR describes the features implemented by the node that owns error record <n>. <q> 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 q = n.

This register is present only when $n == 0$ or ($n \geq 1, n \leq 6$ and $\text{RAM_PROTECTION} == 1$). Otherwise, direct accesses to $\text{ERR}\langle n \rangle\text{STATUS}$ are UNDEFINED.

Attributes

Width

64

Component

RAS

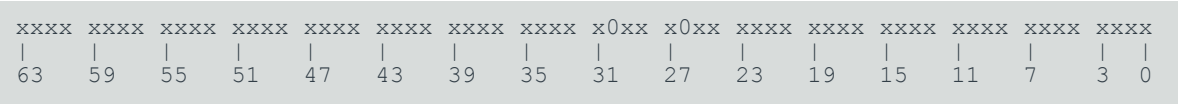
Register offset

$0x010 + (64 * n)$

Access type

inconsistent

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-5: $\text{EXT_ERR}\langle n \rangle\text{STATUS}$ bit assignments

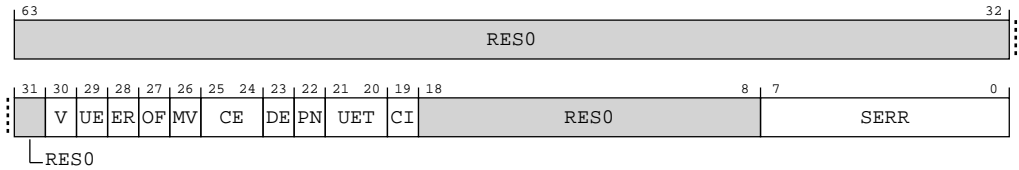


Table B-12: $\text{ERR}\langle n \rangle\text{STATUS}$ bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0
[30]	V	Status Register Valid. 0b0 ERR<n>STATUS not valid. 0b1 ERR<n>STATUS valid. At least one error has been recorded. Access to this field is: W1C	0b0

Bits	Name	Description	Reset
[29]	UE	<p>Uncorrected Error.</p> <p>0b0</p> <p>No errors have been detected, or all detected errors have been either corrected or deferred.</p> <p>0b1</p> <p>At least one detected error was not corrected and not deferred.</p> <p>When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0'</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: W1C</p>	x
[28]	ER	<p>Error Reported.</p> <p>0b0</p> <p>No in-band error response (External Abort) signaled to the Requester making the access or other transaction.</p> <p>0b1</p> <p>An in-band error response was signaled by the component to the Requester making the access or other transaction. This can be because any of the following are true:</p> <ul style="list-style-type: none"> The ERRCTLR[FirstRecordOfNode(n)].UE field, or applicable one of the ERRCTLR[FirstRecordOfNode(n)].{WUE, RUE} fields, is implemented and was 1 when an error was detected and not corrected. The ERRCTLR[FirstRecordOfNode(n)].{WUE, RUE, UE} fields are not implemented and the component always reports errors. <p>Deferred error will not set this field to 1. When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0' or ext-ERR<n>STATUS.UE == '0'</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: W1C</p>	x

Bits	Name	Description	Reset
[27]	OF	<p>Overflow.</p> <p>Indicates that multiple errors have been detected. This field is set to 1 when one of the following occurs:</p> <ul style="list-style-type: none"> A Corrected error counter is implemented, an error is counted, and the counter overflows. ERR<n>STATUS.V was previously 1, a Corrected error counter is not implemented, and a Corrected error is recorded. ERR<n>STATUS.V was previously 1, and a type of error other than a Corrected error is recorded. <p>Otherwise, this field is unchanged when an error is recorded.</p> <p>If a Corrected error counter is implemented, then:</p> <ul style="list-style-type: none"> A direct write that modifies the counter overflow flag indirectly might set this field to an UNKNOWN value. A direct write to this field that clears this field to zero might indirectly set the counter overflow flag to an UNKNOWN value. <p>0b0</p> <p>Since this field was last cleared to zero, no error syndrome has been discarded and, if a Corrected error counter is implemented, it has not overflowed.</p> <p>0b1</p> <p>Since this field was last cleared to zero, at least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed.</p> <p>When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0'</p> <p>Access to this field is: UNKNOWN/W1</p> <p>Otherwise</p> <p>Access to this field is: W1C</p>	x
[26]	MV	<p>Miscellaneous Registers Valid.</p> <p>0b0</p> <p>ERR<n>MISC<m> not valid.</p> <p>0b1</p> <p>The contents of the ERR<n>MISC<m> registers contain additional information for an error recorded by this record.</p> <p>Access to this field is: W1C</p>	0b0

Bits	Name	Description	Reset
[25:24]	CE	<p>Corrected Error.</p> <p>0b00 No errors were corrected.</p> <p>0b10 At least one error was corrected.</p> <p>When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0' Access to this field is: UNKNOWN/WI</p> <p>Otherwise Access to this field is: W1C</p>	xx
[23]	DE	<p>Deferred Error.</p> <p>0b0 No errors were deferred.</p> <p>0b1 At least one error was not corrected and deferred.</p> <p>When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0' Access to this field is: UNKNOWN/WI</p> <p>Otherwise Access to this field is: W1C</p>	x
[22]	PN	<p>Poison.</p> <p>0b0 Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC), or Corrected error recorded.</p> <p>0b1 Uncorrected error or Deferred error recorded because a poison value was detected.</p> <p>When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0' or ext-ERR<n>STATUS.[DE,UE] == '00' Access to this field is: UNKNOWN/WI</p> <p>Otherwise Access to this field is: W1C</p>	x

Bits	Name	Description	Reset
[21:20]	UET	<p>Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error.</p> <p>0b00 Uncorrected error, Uncontainable error (UC).</p> <p>0b01 Uncorrected error, Unrecoverable error (UEU).</p> <p>0b10 Uncorrected error, Latent or Restartable error (UEO).</p> <p>0b11 Uncorrected error, Signaled or Recoverable error (UER).</p> <p>UER can mean either Signaled or Recoverable error, and UEO can mean either Latent or Restartable error.</p> <p>When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0' or ext-ERR<n>STATUS.UE == '0' Access to this field is: UNKNOWN/WI</p> <p>Otherwise Access to this field is: W1C</p>	xx
[19]	CI	<p>Critical Error. Indicates whether a critical error condition has been recorded.</p> <p>0b0 No critical error condition.</p> <p>0b1 Critical error condition.</p> <p>When clearing ERR<n>STATUS.V to 0, if this field is nonzero, then Arm recommends that software write 1 to this field to clear this field to zero.</p> <p>When ext-ERR<n>STATUS.V == '0' Access to this field is: UNKNOWN/WI</p> <p>Otherwise Access to this field is: W1C</p>	x
[18:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	SERR	<p>Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry.</p> <p>0b00000000 No error.</p> <p>0b00000010 Data value from (non-associative) internal memory. For example, ECC from on-chip SRAM or buffer.</p> <p>0b00000110 Data value from associative memory. For example, ECC error on cache data.</p> <p>0b00000111 Address/control value from associative memory. For example, ECC error on cache tag.</p> <p>0b00001000 Data value from a TLB. For example, ECC error on TLB data.</p> <p>0b00001001 Address/control value from a TLB. For example, ECC error on TLB tag.</p> <p>0b00010010 Error response from Completer of access. For example, error response from cache write-back.</p> <p>This value applies when CHI == 1.</p> <p>0b00010101 Deferred error from Completer not supported at Requester. For example, poisoned data received from the Completer of an access by a Requester that cannot defer the error further.</p> <p>All other values are reserved.</p> <p>When ext-ERR<n>STATUS.V == '0' Access to this field is: UNKNOWN/WI</p> <p>When ext-ERR<n>STATUS.DE == '0', ext-ERR<n>STATUS.UE == '0', ext-ERR<n>STATUS.CE != '00' and ERR<n>STATUS.CE is not being cleared to 0b00 in the same write Access to this field is: RO</p> <p>When ext-ERR<n>STATUS.UE == '0', ext-ERR<n>STATUS.DE != '0' and ERR<n>STATUS.DE is not being cleared to 0b0 in the same write Access to this field is: RO</p> <p>When ext-ERR<n>STATUS.UE != '0' and ERR<n>STATUS.UE is not being cleared to 0b0 in the same write Access to this field is: RO</p> <p>Otherwise Access to this field is: RW</p>	8 {x}

Access

ERR<n>STATUS.{AV, V, UE, ER, OF, MV, CE, DE, PN, UET, CI} are write-one-to-clear (W1C) fields, meaning writes of zero are ignored, and a write of one or all-ones to the field clears the field to zero. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is **IMPLEMENTATION DEFINED**. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, software must clear the valid fields in the register to allow new errors to be recorded. However, between reading the register and clearing the valid fields, a new

error might have overwritten the register. To prevent this error being lost by software, the register prevents updates to fields that might have been updated by a new error.

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, a write to ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as **UNKNOWN** values when ERR<n>STATUS.V is zero, this means those fields can be implemented as read-only if applicable.

To ensure correct and portable operation, when software is clearing the valid fields in the register to allow new errors to be recorded, Arm recommends that software:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- Write ones to all the W1C fields that are nonzero in the read value.
- Write zero to all the W1C fields that are zero in the read value.
- Write zero to all the RW fields.

Otherwise, these fields might not have the correct value when a new fault is recorded.

An exception is when the node supports writing to these fields as part of fault injection. See also ext-ERR<n>PFGF.SYN.

Accessibility

ERR<n>STATUS.{AV, V, UE, ER, OF, MV, CE, DE, PN, UET, CI} are write-one-to-clear (W1C) fields, meaning writes of zero are ignored, and a write of one or all-ones to the field clears the field to zero. ERR<n>STATUS.{IERR, SERR} are read/write (RW) fields, although the set of implemented valid values is IMPLEMENTATION DEFINED. See also ext-ERR<n>PFGF.SYN.

After reading ERR<n>STATUS, software must clear the valid fields in the register to allow new errors to be recorded. However, between reading the register and clearing the valid fields, a new error might have overwritten the register. To prevent this error being lost by software, the register prevents updates to fields that might have been updated by a new error.

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, a write to ERR<n>STATUS when ERR<n>STATUS.V is 1 results in either ERR<n>STATUS.V field being cleared to zero, or ERR<n>STATUS.V not changing. Since all fields in ERR<n>STATUS, other than ERR<n>STATUS.{AV, V, MV}, usually read as UNKNOWN values when ERR<n>STATUS.V is zero, this means those fields can be implemented as read-only if applicable.

To ensure correct and portable operation, when software is clearing the valid fields in the register to allow new errors to be recorded, Arm recommends that software:

- Read ERR<n>STATUS and determine which fields need to be cleared to zero.
- Write ones to all the W1C fields that are nonzero in the read value.
- Write zero to all the W1C fields that are zero in the read value.
- Write zero to all the RW fields.

Otherwise, these fields might not have the correct value when a new fault is recorded.

An exception is when the node supports writing to these fields as part of fault injection. See also ext-ERR<n>PFGF.SYN.

Component	Offset	Instance	Range
RAS	0x010 + (64 * n)	ERR<n>STATUS	None

This interface is accessible as follows:

When ext-ERR<n>STATUS.V != '0' and ERR<n>STATUS.V is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.UE != '0' and ERR<n>STATUS.UE is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.OF != '0' and ERR<n>STATUS.OF is not being cleared to 0b0 in the same write

RO

When ext-ERR<n>STATUS.CE != '00' and ERR<n>STATUS.CE is not being cleared to 0b00 in the same write

RO

When ext-ERR<n>STATUS.DE != '0' and ERR<n>STATUS.DE is not being cleared to 0b0 in the same write

RO

Otherwise

RW

B.1.2.1.4 ERR<n>MISCO, Error Record <n> Miscellaneous Register 0, n = 0 - 9

Error syndrome register, containing corrected error counter, FRU identification and other fault state information.

Configurations

1 or 7 error records are implemented, depending on the RAM_PROTECTION parameter.

If error record <n> is not implemented, ERR<n>MISCO is RES0.

ERR<q>FR describes the features implemented by the node that owns error record <n>. <q> 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 q = n.

Attributes

Width

64

Component

RAS

Register offset

0x020 + (64 * n)

Access type

Read

R

Write

W

Reset value

When n == 0

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

When n >= 1, n <= 3 and RAM_PROTECTION == 1

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

When n >= 4, n <= 6 and RAM_PROTECTION == 1

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 n == 0

Figure B-6: EXT_ERR<n>MISC0 bit assignments

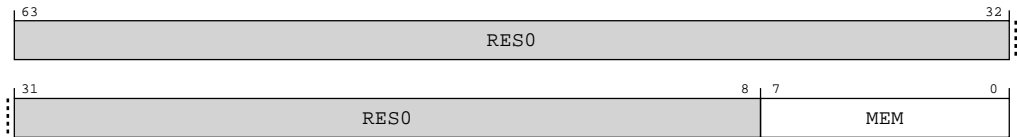


Table B-14: ERR<n>MISC0 bit descriptions

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	RES0
[7:0]	MEM	<div>Indicates which memory encountered the error.</div> <div>0b01000000 Eviction of poisoned cache line to MM, but the port does not support poison. This value applies when CHI == 0.</div> <div>0b01000011 Non-data error response received by MM while attempting an L2 cache line writeback or eviction. The error can be SLVERR or DECERR, if the MM port is configured as AXI, or NDERR if the MM port is configured as CHI. Evictions caused by a Set/Way maintenance operation will return the error to the core and not report the error to RAS.</div> <div>0b01000100 Data error response received by MM with PassDirty attribute set, but the processor is not configured with ECC support, or ECC checking is currently disabled. This occurs when the MM port is configured as CHI and the error is DERR. This value applies when CHI == 1.</div> <div>0b00100001 Simultaneous MM errors of above types.</div> <div>Other values are Reserved.</div>	8 {x}

When n >= 1, n <= 3 and RAM_PROTECTION == 1

Figure B-7: EXT_ERR<n>MISC0 bit assignments

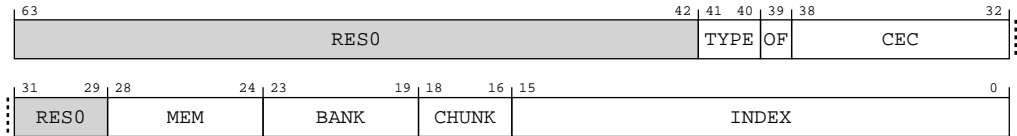


Table B-15: ERR<n>MISC0 bit descriptions

Bits	Name	Description	Reset
[63:42]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[41:40]	TYPE	Error type. 0b00 SECEDED scheme detected 1-bit error. 0b01 SECEDED scheme detected 2-bit error. 0b10 SECEDED scheme detected 1-bit or 2-bit error.	xx
[39]	OF	Sticky overflow bit. Set to 1 when the Corrected error count field is incremented and wraps through zero. 0b0 Counter has not overflowed. 0b1 Counter has overflowed. A direct write that modifies this bit might indirectly set ext-ERR<n>STATUS.OF to an UNKNOWN value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.	x
[38:32]	CEC	Corrected error count. Incremented for each Corrected error. Deferred and Uncorrected errors are not counted.	7 {x}
[31:29]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[28:24]	MEM	<p>Indicates which memory encountered the error.</p> <p>0b00001 L1 I-cache data error.</p> <p>0b00010 L1 I-cache tag error.</p> <p>0b00011 L1 I-cache simultaneous tag and data errors.</p> <p>0b01001 L1 D-cache data error.</p> <p>0b01010 L1 D-cache tag error.</p> <p>0b01011 L1 D-cache simultaneous tag and data errors.</p> <p>0b01100 L1 D-cache dirty error.</p> <p>0b01101 L1 D-cache simultaneous data and dirty errors.</p> <p>0b01110 L1 D-cache simultaneous tag and dirty errors.</p> <p>0b01111 L1 D-cache simultaneous tag, data and dirty errors.</p> <p>0b10000 ITCM error.</p> <p>0b10001 DTCM error.</p> <p>0b11001 MMS error data error.</p> <p>0b11010 MMS error tag error.</p> <p>0b11011 MMS error simultaneous tag and data errors.</p> <p>Other values are Reserved.</p>	5{x}
[23:19]	BANK	<p>Indicates which bank within the memory encountered the error.</p> <p>Values in the range 0b00000 - 0b01111 (0x00 - 0x0F) indicate a single bank within the memory specified by the MEM field.</p> <p>Value 0b11111 (0x1F) indicates that simultaneous errors were detected within the memory specified by the MEM field.</p> <p>Other values are Reserved.</p>	5{x}

Bits	Name	Description	Reset
[18:16]	CHUNK	Indicates which ECC chunk within the bank encountered the error. Values in the range 0b000 - 0b011 (0x0 - 0x3) indicate a single ECC chunk within the bank specified by the BANK field. Value 0b111 (0x7) indicates that simultaneous errors were detected within the memory specified by the MEM field. Other values are Reserved.	xxx
[15:0]	INDEX	Indicates what was the memory address being accessed when the error occurred. When the MEM field indicates a cache tag, data or dirty memory, the INDEX field indicates the cache line index which was being accessed. When the MEM field indicates ITCM, DTCM, MMS or the L2 buffers, the INDEX field indicates the memory address which was being accessed. Value 0xFF indicates that simultaneous errors were detected within the memory specified by the MEM field.	16{x}

When $n \geq 4$, $n \leq 6$ and $\text{RAM_PROTECTION} == 1$

Figure B-8: EXT_ERR<n>MISC0 bit assignments

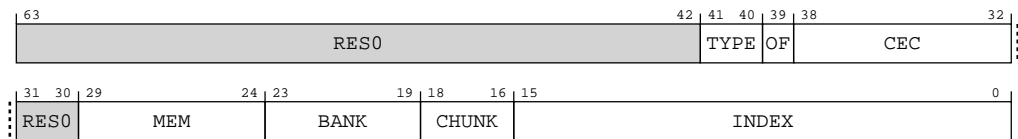


Table B-16: ERR<n>MISC0 bit descriptions

Bits	Name	Description	Reset
[63:42]	RES0	Reserved	RES0
[41:40]	TYPE	Error type. 0b00 SECEDED scheme detected 1-bit error. 0b01 SECEDED scheme detected 2-bit error. 0b10 SECEDED scheme detected 1-bit or 2-bit error.	xx

Bits	Name	Description	Reset
[39]	OF	<p>Sticky overflow bit.</p> <p>Set to 1 when the Corrected error count field is incremented and wraps through zero.</p> <p>0b0 Counter has not overflowed.</p> <p>0b1 Counter has overflowed.</p> <p>A direct write that modifies this bit might indirectly set ext-ERR<n>STATUS.OF to an UNKNOWN value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.</p>	x
[38:32]	CEC	<p>Corrected error count.</p> <p>Incremented for each Corrected error. Deferred and Uncorrected errors are not counted.</p>	7 {x}
[31:30]	RES0	Reserved	RES0
[29:24]	MEM	<p>Indicates which memory encountered the error.</p> <p>0b100000 L2 cache tag error.</p> <p>0b100001 L2 cache simultaneous errors in two or more of the following: L2 tags, duplicate L1 tags, data or buffers.</p> <p>0b100010 L2 cache data error.</p> <p>0b100011 L2 cache buffers error.</p> <p>0b101000 L2 cache core 0 duplicate L1 tag error.</p> <p>0b101001 L2 cache core 1 duplicate L1 tag error.</p> <p>0b101010 L2 cache core 2 duplicate L1 tag error.</p> <p>0b101011 L2 cache core 3 duplicate L1 tag error.</p> <p>0b101100 L2 cache core 4 duplicate L1 tag error.</p> <p>0b101101 L2 cache core 5 duplicate L1 tag error.</p> <p>0b101110 L2 cache core 6 duplicate L1 tag error.</p> <p>0b101111 L2 cache core 7 duplicate L1 tag error.</p>	6 {x}

Bits	Name	Description	Reset
[29:24 continued]	MEM	<p>0b110000 LCU core 0 duplicate L1 tag error.</p> <p>0b110001 LCU core 1 duplicate L1 tag error.</p> <p>0b110010 LCU core 2 duplicate L1 tag error.</p> <p>0b110011 LCU core 3 duplicate L1 tag error.</p> <p>0b110100 LCU core 4 duplicate L1 tag error.</p> <p>0b110101 LCU core 5 duplicate L1 tag error.</p> <p>0b110110 LCU core 6 duplicate L1 tag error.</p> <p>0b110111 LCU core 7 duplicate L1 tag error.</p> <p>0b111000 LCU simultaneous errors in two or more of the duplicate L1 tags.</p> <p>Other values are Reserved.</p>	6{x}
[23:19]	BANK	<p>Indicates which bank within the memory encountered the error.</p> <p>Values in the range 0b00000 - 0b01111 (0x00 - 0x0F) indicate a single bank within the memory specified by the MEM field.</p> <p>Value 0b11111 (0x1F) indicates that simultaneous errors were detected within the memory specified by the MEM field.</p> <p>Other values are Reserved.</p>	5{x}
[18:16]	CHUNK	<p>Indicates which ECC chunk within the bank encountered the error.</p> <p>Values in the range 0b000 - 0b011 (0x0 - 0x3) indicate a single ECC chunk within the bank specified by the BANK field.</p> <p>Value 0b111 (0x7) indicates that simultaneous errors were detected within the memory specified by the MEM field.</p> <p>Other values are Reserved.</p>	xxx
[15:0]	INDEX	<p>Indicates what was the memory address being accessed when the error occurred.</p> <p>When the MEM field indicates a cache tag, data or dirty memory, the INDEX field indicates the cache line index which was being accessed.</p> <p>When the MEM field indicates ITCM, DTCM, MMS or the L2 buffers, the INDEX field indicates the memory address which was being accessed.</p> <p>Value 0xFF indicates that simultaneous errors were detected within the memory specified by the MEM field.</p>	16{x}

Accessibility

When ext-ERR<n>STATUS.MV is set to 1, the miscellaneous syndrome for the most recently recorded error should ignore writes.

Component	Offset	Instance	Range
RAS	0x020 + (64 * n)	ERR<n>MISC0	None

This interface is accessible as follows:

RW

B.1.2.1.5 ERR<n>MISC1, Error Record <n> Miscellaneous Register 1, n = 0 - 9

Reserved error syndrome register.

Configurations

This register is present only when IsErrorRecordImplemented(n). Otherwise, direct accesses to ERR<n>MISC1 are UNDEFINED.

Attributes

Width

64

Component

RAS

Register offset

0x028 + (64 * n)

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-9: EXT_ERR<n>MISC1 bit assignments

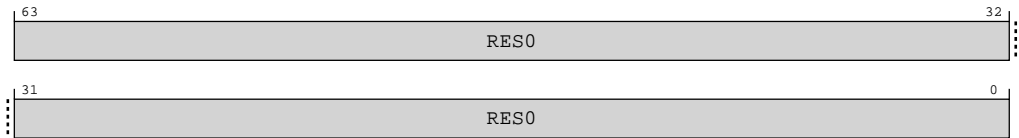


Table B-18: ERR<n>MISC1 bit descriptions

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.



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.



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	0x028 + (64 * n)	ERR<n>MISC1	None

This interface is accessible as follows:

RW

B.1.2.1.6 ERR<n>MISC2, Error Record <n> Miscellaneous Register 2, n = 0 - 9

Reserved error syndrome register.

Configurations

This register is present only when IsErrorRecordImplemented(n). Otherwise, direct accesses to ERR<n>MISC2 are UNDEFINED.

Attributes

Width

64

Component

RAS

Register offset

0x030 + (64 * n)

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-10: EXT_ERR<n>MISC2 bit assignments

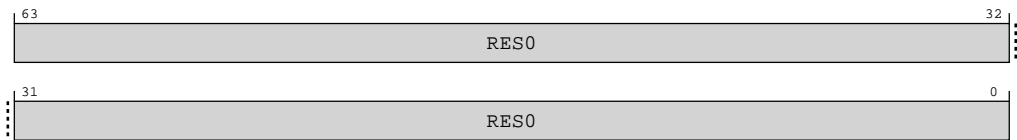


Table B-20: ERR<n>MISC2 bit descriptions

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.



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	0x030 + (64 * n)	ERR<n>MISC2	None

This interface is accessible as follows:

RW

B.1.2.1.7 ERR<n>MISC3, Error Record <n> Miscellaneous Register 3, n = 0 - 9

Reserved error syndrome register.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

RAS

Register offset

0x038 + (64 * n)

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-11: EXT_ERR<n>MISC3 bit assignments

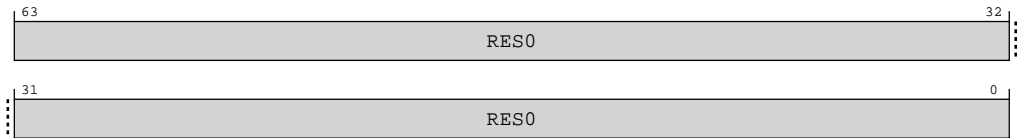


Table B-22: ERR<n>MISC3 bit descriptions

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

Access

Arm recommends that a miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.

When ext-ERR<n>STATUS.MV is set to 1, the miscellaneous syndrome for the most recently recorded error should ignore writes.

Accessibility

Arm recommends that a miscellaneous syndrome for multiple errors, such as a corrected error counter, is read/write.

When ext-ERR<n>STATUS.MV is set to 1, the miscellaneous syndrome for the most recently recorded error should ignore writes.

Component	Offset	Instance	Range
RAS	0x038 + (64 * n)	ERR<n>MISC3	None

This interface is accessible as follows:

RW

B.1.2.1.8 ERR<n>PFGF, Error Record <n> Pseudo-fault Generation Feature Register, n = 0 - 9

Defines which common architecturally-defined fault generation features are implemented.

Configurations

Present only when error record <n> is implemented, and error record <n> is the first error record owned by a node. Otherwise, RES0.

1 or 7 error records are implemented, depending on the RAM_PROTECTION parameter.

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

$$0x800 + (64 * n)$$

Access type

RO

Reset value

When (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx x100 xxxx xxxx xxxx xxx0 0000 xxx0 0010

XXXX XXXX XXXX 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 (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

Figure B-12: EXT_ERR<n>PFGF bit assignments

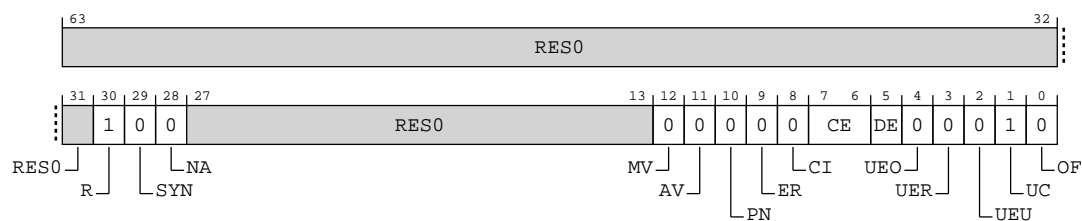
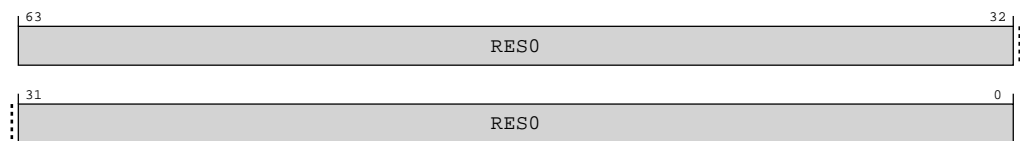


Table B-24: ERR<n>PFGF bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	RES0
[30]	R	<p>Restartable. Support for Error Generation Counter restart mode.</p> <p>0b1</p> <p>Error Generation Counter restart mode is implemented and is controlled by ext-ERR<n>PFGCTL.R. ext-ERR<n>PFGCTL.R is a read/write field.</p>	0b1
[29]	SYN	<p>Syndrome. Fault syndrome injection.</p> <p>0b0</p> <p>When an injected error is recorded, the node sets ext-ERR<n>STATUS.{IERR, SERR} to IMPLEMENTATION DEFINED values. ext-ERR<n>STATUS.{IERR, SERR} are UNKNOWN when ext-ERR<n>STATUS.V is 0.</p> <p>Note:</p> <p>If ERR<n>PFGF.SYN is 1, software can write specific values into the ext-ERR<n>STATUS.{IERR, SERR} fields when setting up a fault injection event. The sets of values that can be written to these fields is IMPLEMENTATION DEFINED.</p>	0b0
[28]	NA	<p>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.</p> <p>0b0</p> <p>The component fakes detection of the error on an access to the component.</p>	0b0
[27:13]	RES0	Reserved	RES0
[12]	MV	<p>Miscellaneous syndrome.</p> <p>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.</p> <p>0b0</p> <p>When an injected error is recorded, the node will record additional syndrome in ERR<n>MISCO and ext-ERR<n>STATUS.MV will be set to 1.</p> <p>If ERR<n>PFGF.MV is 1, software can write specific additional syndrome values into the ERR<n>MISC<m> registers when setting up a fault injection event. The permitted values that can be written to these registers are IMPLEMENTATION DEFINED.</p>	0b0
[11]	AV	<p>Address syndrome. Address syndrome injection.</p> <p>0b0</p> <p>When an injected error is recorded, the node leaves ext-ERR<n>ADDR unchanged. ext-ERR<n>STATUS.AV is always set to 0.</p>	0b0
[10]	PN	<p>Poison flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.PN status flag.</p> <p>0b0</p> <p>When an injected error is recorded, the node does not set the ext-ERR<n>STATUS.PN bit. ext-ERR<n>PFGCTL.PN is RES0.</p> <p>This behavior replaces the architecture-defined rules for setting the ext-ERR<n>STATUS.PN bit.</p>	0b0
[9]	ER	<p>Error Reported flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.ER status flag.</p> <p>0b0</p> <p>When an injected error is recorded, the node sets ext-ERR<n>STATUS.ER according to the architecture-defined rules for setting the ER field. ext-ERR<n>PFGCTL.ER is RES0.</p>	0b0

Bits	Name	Description	Reset
[8]	CI	<p>Critical Error flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.CI status flag.</p> <p>0b0</p> <p>When an injected error is recorded with ext-ERR<n>PFGCTL.UC == 0b1, the node sets ext-ERR<n>STATUS.CI to 1 in the following cases:</p> <ul style="list-style-type: none"> When a snoop request is received by the L1 D-cache. When L2 cache is accessed. <p>This behavior replaces the architecture-defined rules for setting the ext-ERR<n>STATUS.CI bit. ext-ERR<n>PFGCTL.CI is RES0.</p>	0b0
[7:6]	CE	<p>When n == 1 or n == 4</p> <p>Corrected Error generation. Describes the types of Corrected Error that the fault generation feature of the node can generate.</p> <p>01</p> <p>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. ext-ERR<n>PFGCTL.CE is a read/write field. The values 0b10 and 0b11 in ext-ERR<n>PFGCTL.CE are reserved.</p> <p>Otherwise</p> <p>RES0</p>	xx
[5]	DE	<p>When n == 1 or n == 4</p> <p>Deferred Error generation. Describes whether the fault generation feature of the node can generate Deferred errors.</p> <p>1</p> <p>The fault generation feature of the node allows generation of Deferred errors. ext-ERR<n>PFGCTL.DE is a read/write field.</p> <p>Otherwise</p> <p>Deferred Error generation. Describes whether the fault generation feature of the node can generate Deferred errors.</p> <p>0</p> <p>The fault generation feature of the node does not generate Deferred errors. ext-ERR<n>PFGCTL.DE is RES0.</p>	x
[4]	UEO	<p>Latent or Restartable Error generation. Describes whether the fault generation feature of the node can generate Latent or Restartable errors.</p> <p>0b0</p> <p>The fault generation feature of the node does not generate Latent or Restartable errors. ext-ERR<n>PFGCTL.UEO is RES0.</p>	0b0
[3]	UER	<p>Signaled or Recoverable Error generation. Describes whether the fault generation feature of the node can generate Signaled or Recoverable errors.</p> <p>0b0</p> <p>The fault generation feature of the node does not generate Signaled or Recoverable errors. ext-ERR<n>PFGCTL.UER is RES0.</p>	0b0

Bits	Name	Description	Reset
[2]	UEU	Unrecoverable Error generation. Describes whether the fault generation feature of the node can generate Unrecoverable errors. 0b0 The fault generation feature of the node does not generate Unrecoverable errors. ext-ERR<n>PFGCTL.UEU is RES0 .	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. ext-ERR<n>PFGCTL.UC is a read/write field.	0b1
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.OF status flag. 0b0 When an injected error is recorded, the node sets ext-ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF field. ext-ERR<n>PFGCTL.OF is RES0 .	0b0

Figure B-13: EXT_ERR<n>PFGF bit assignments**Table B-25: ERR<n>PFGF bit descriptions**

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

Accessibility

Component	Offset	Instance	Range
RAS	0x800 + (64 * n)	ERR<n>PFGF	None

This interface is accessible as follows:

RO

B.1.2.1.9 ERR<n>PFGCTL, Error Record <n> Pseudo-fault Generation Control Register, n = 0 - 9

Enables controlled fault generation.

Configurations

Present only when error record <n> is implemented, and error record <n> is the first error record owned by a node. Otherwise, RES0.

1 or 7 error records are implemented, depending on the RAM_PROTECTION parameter.

ext-ERR<n>FR describes the features implemented by the node.

Attributes

Width

64

Component

RAS

Register offset

0x808 + (64 * n)

Access type

RW

Reset value

When (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0xxx xxxx xxxx xxxx xxx1 xxxx xxxx 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 (n == 1 and RAM_PROTECTION == 1) or (n == 4 and RAM_PROTECTION == 1)

Figure B-14: EXT_ERR<n>PFGCTL bit assignments

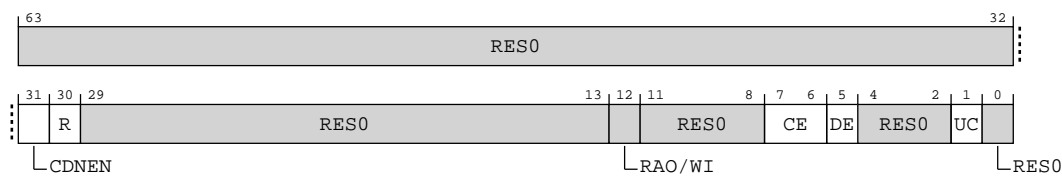


Table B-27: ERR<n>PFGCTL bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	CDNEN	Countdown Enable. Controls transfers of the value held in ext-ERR<n>PFGCDN to the Error Generation Counter and enables this counter. 0b0 The Error Generation Counter is disabled. 0b1 The Error Generation Counter is enabled. On a write of 1 to this field, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN.	0b0

Bits	Name	Description	Reset
[30]	R	Restart. Controls whether, on reaching zero, the Error Generation Counter restarts from the ext-ERR<n>PFGCDN value, or stops. 0b0 On reaching 0, the Error Generation Counter stops. 0b1 On reaching 0, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN.	x
[29:13]	RES0	Reserved	RES0
[12]	RAO/WI	Reserved	RAO/ WI
[11:8]	RES0	Reserved	RES0
[7:6]	CE	When n == 1 or n == 4 Corrected Error generation enable. Controls the type of injected Corrected error generated by the fault injection feature of the node. 0b00 An injected Corrected error will not be generated by the fault injection feature of the node. 0b01 An injected non-specific Corrected error is generated in the fault injection state. ext-ERR<n>STATUS.CE is set to 0b10 when the injected error is recorded. The set of permitted values for this field is defined by ext-ERR<n>PFGF.CE. The node enters the fault injection state when the Error Generation Counter decrements to zero. The injected error is not generated when the error is generated on an access to the component in the fault injection state and the data is not consumed. Otherwise RES0	xx
[5]	DE	When n == 1 or n == 4 Deferred Error generation enable. Controls whether an injected Deferred error is generated by the fault injection feature of the node. 0b0 An injected Deferred error will not be generated by the fault generation feature of the node. 0b1 An injected Deferred error is generated in the fault injection state. The node enters the fault injection state when the Error Generation Counter decrements to zero. The injected error is not generated when the error is generated on an access to the component in the fault injection state and the data is not consumed. Otherwise RES0	x
[4:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1]	UC	Uncontainable Error generation enable. Controls whether an injected Uncontainable error is generated by the fault injection feature of the node. 0b0 An injected Uncontainable error will not be generated by the fault generation feature of the node. 0b1 An injected Uncontainable error is generated in the fault injection state. The node enters the fault injection state when the Error Generation Counter decrements to zero. The injected error is not generated when the error is generated on an access to the component in the fault injection state and the data is not consumed.	x
[0]	RES0	Reserved	RES0

Figure B-15: EXT_ERR<n>PFGCTL bit assignments

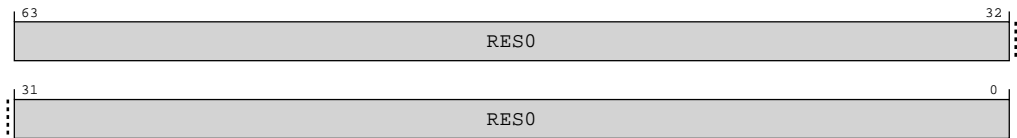


Table B-28: ERR<n>PFGCTL bit descriptions

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

Accessibility

Component	Offset	Instance	Range
RAS	0x808 + (64 * n)	ERR<n>PFGCTL	None

This interface is accessible as follows:

RW

B.1.2.1.10 ERR<n>PFGCDN, Error Record <n> Pseudo-fault Generation Countdown Register, n = 0 - 9

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.

This register is present only when (n == 1 or n == 4) and RAM_PROTECTION == 1. Otherwise, direct accesses to ERR<n>PFGCDN are UNDEFINED.

Attributes

Width

64

Component

RAS

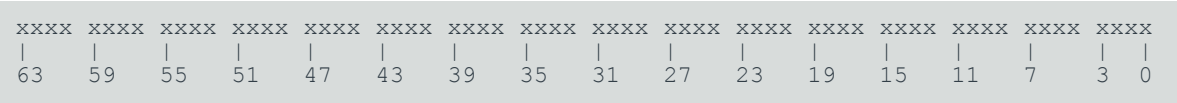
Register offset

0x810 + (64 * n)

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-16: EXT_ERR<n>PFGCDN bit assignments

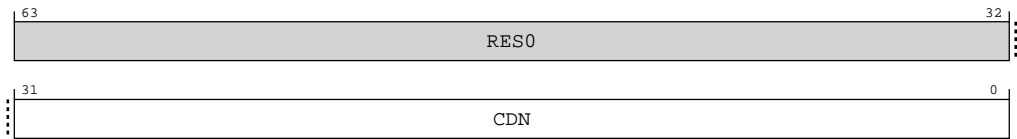


Table B-30: ERR<n>PFGCDN bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	CDN	<p>Countdown value.</p> <p>This field is copied to Error Generation Counter when either:</p> <ul style="list-style-type: none">Software writes 1 to ext-ERR<n>PFGCTL.CDNEN.The Error Generation Counter decrements to zero and ext-ERR<n>PFGCTL.R is 1. <p>While ext-ERR<n>PFGCTL.CDNEN is 1 and the Error Generation Counter is nonzero, the counter decrements by 1 for each cycle at the core clock rate. When the counter reaches zero, one of the errors enabled in the ext-ERR<n>PFGCTL register is generated.</p> <p>Note: The current Error Generation Counter value is not visible to software.</p>	32 {x}

Accessibility

Component	Offset	Instance	Range
RAS	0x810 + (64 * n)	ERR<n>PFGCDN	None

This interface is accessible as follows:

RW

B.1.2.1.11 ERRGSR, Error Group Status Register

Shows the status for the records in the group.

Configurations

1 or 7 error records are implemented, depending on the RAM_PROTECTION parameter.

If an error record is not implemented, the corresponding bit in this register will read as 0.

Attributes

Width

64

Component

RAS

Register offset

0xE00

Access type

RO

Reset value

xxxx	xxxx	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	00xx	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-17: EXT_ERRGSR bit assignments

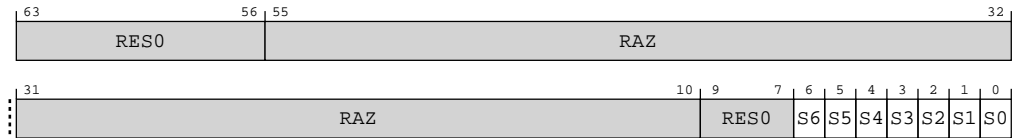


Table B-32: ERRGSR bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	RES0
[55:10]	RAZ	Reserved	RAZ
[9:7]	RES0	Reserved	RES0
[6]	S6	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. 0b0 No error. 0b1 One or more memory ECC errors from L2 cache or LCU. See ext-ERR6STATUS.V.	x
[5]	S5	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. 0b0 No error. 0b1 One or more memory ECC errors from L2 cache or LCU. See ext-ERR5STATUS.V.	x
[4]	S4	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. 0b0 No error. 0b1 One or more memory ECC errors from L2 cache or LCU. See ext-ERR4STATUS.V.	x
[3]	S3	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. 0b0 No error. 0b1 One or more memory ECC errors from L1 I-cache, L1 D-cache, ITCM, DTCM or MMS. See ext-ERR3STATUS.V.	x
[2]	S2	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. 0b0 No error. 0b1 One or more memory ECC errors from L1 I-cache, L1 D-cache, ITCM, DTCM or MMS. See ext-ERR2STATUS.V.	x

Bits	Name	Description	Reset
[1]	S1	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. 0b0 No error. 0b1 One or more memory ECC errors from L1 I-cache, L1 D-cache, ITCM, DTCM or MMS. See ext-ERR1STATUS.V.	x
[0]	S0	The status for error record <m>. A read-only copy of ERR<m>STATUS.V. 0b0 No error. 0b1 One or more memory non-ECC errors from L2 cache or LCU. See ext-ERR0STATUS.V.	x

Accessibility

Component	Offset	Instance	Range
RAS	0xE00	ERRGSR	None

This interface is accessible as follows:

RO

B.1.2.1.12 ERRIIDR, Implementation Identification Register

Defines the implementer of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

RAS

Register offset

0xE10

Access type

RO

Reset value

1101	0001	0101	0111	xxxx	0100	x011	1011
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-18: EXT_ERRIIDR bit assignments

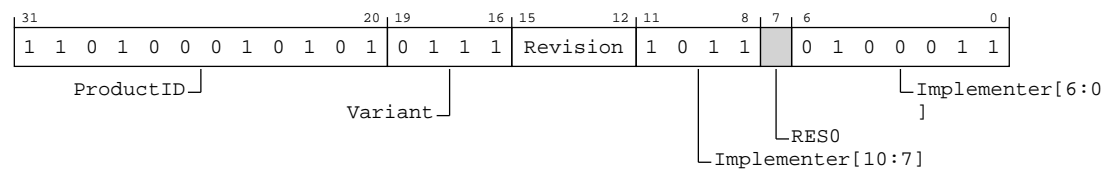


Table B-34: ERRIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	Part number. 0b110100010101 Cortex-R82 RAS component.	0xD15
[19:16]	Variant	Component major revision. 0b0111 Product revision 7.	0b0111
[15:12]	Revision	Component minor revision. 0b0000 No ECO fixes.	xxxx
[11:8, 6:0]	Implementer	Contains the JEP106 code of the company that implemented the RAS component. 0b01000111011 Arm Limited. Bits [11:8] contain the JEP106 continuation code of the implementer, and bits [6:0] contain the JEP106 identity code of the implementer. ext-ERRPIDR4.DES_2 matches bits [11:8] of ERRIIDR.Implementer, ext-ERRPIDR2.DES_1 matches bits [6:4] of ERRIIDR.Implementer, and ext-ERRPIDR1.DES_0 matches bits [3:0] of ERRIIDR.Implementer.	0b01000111011
[7]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
RAS	0xE10	None

This interface is accessible as follows:

RO

B.1.2.1.13 ERRDEVAFF, Device Affinity Register

The processor records have mixed affinity, and so this register is not implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

RAS

Register offset

0xFA8

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
63	59	55	51	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_ERRDEVAFF bit assignments

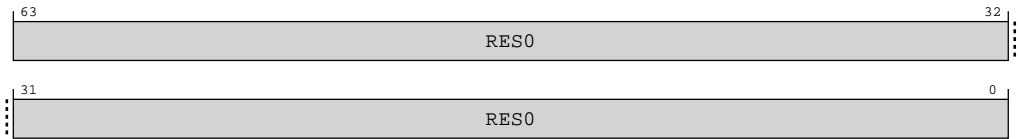


Table B-36: ERRDEVAFF bit descriptions

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

Accessibility

Component	Offset	Instance	Range
RAS	0xFA8	ERRDEVAFF	None

This interface is accessible as follows:

RO

B.1.2.1.14 ERRDEVARCH, Device Architecture Register

Provides discovery information for the component.

Configurations

ERRDEVARCH is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 0001 0000 1010 0000 0000

Bit descriptions

Figure B-20: EXT_ERRDEVARCH bit assignments

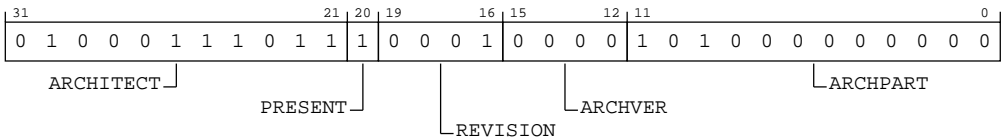


Table B-38: ERRDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. 0b01000111011 JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	0b01000111011
[20]	PRESENT	DEVARCH present. Defines that ERRDEVARCH register is present. Defined values are: 0b1 Device Architecture information present. This field reads as 1.	0b1

Bits	Name	Description	Reset
[19:16]	REVISION	Revision. Defines the architecture revision of the component. Defined values are: 0b0001 RAS System Architecture, error record group v1.1. As 0b0000 and also: <ul style="list-style-type: none"> • Simplifies ext-ERR<n>STATUS. • Adds support for additional ERR<n>MISC<m> registers. • Adds support for the optional RAS Timestamp Extension. • Adds support for the optional Common Fault Injection Model Extension. All other values are reserved.	0b0001
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. Defined values are: 0b0000 RAS System Architecture, error record group v1. All other values are reserved. ERRDEVARCH.ARCHVER and ERRDEVARCH.ARCHPART are also defined as a single field, ERRDEVARCH.ARCHID, so that ERRDEVARCH.ARCHVER is ERRDEVARCH.ARCHID[15:12].	0b0000
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. Defined values are: 0b101000000000 RAS System Architecture, error record group. ERRDEVARCH.ARCHVER and ERRDEVARCH.ARCHPART are also defined as a single field, ERRDEVARCH.ARCHID, so that ERRDEVARCH.ARCHPART is ERRDEVARCH.ARCHID[11:0].	0xA00

Accessibility

Component	Offset	Instance	Range
RAS	0xFBC	ERRDEVARCH	None

This interface is accessible as follows:

RO

B.1.2.1.15 ERRDEVID, Device Configuration Register

Provides discovery information for the component.

Configurations

ERRDEVID is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFC8

Access type

RO

Reset value

xxxx	xxxx	xx0x	0000	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-21: EXT_ERRDEVID bit assignments

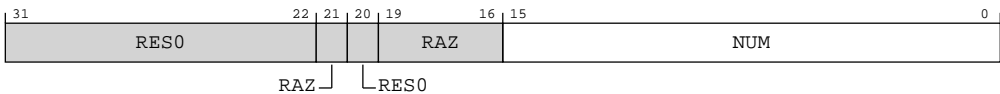


Table B-40: ERRDEVID bit descriptions

Bits	Name	Description	Reset
[31:22]	RES0	Reserved	RES0
[21]	RAZ	Reserved	RAZ
[20]	RES0	Reserved	RES0
[19:16]	RAZ	Reserved	RAZ
[15:0]	NUM	When RAM_PROTECTION == 1 Highest numbered index of the records that can be accessed through the Error Record System registers plus one. 00000000000000111 7 error records implemented. Otherwise Highest numbered index of the records that can be accessed through the Error Record System registers plus one. 0000000000000001 1 error record implemented.	16 {x}

Accessibility

Component	Offset	Instance	Range
RAS	0xFC8	ERRDEVID	None

This interface is accessible as follows:

RO

B.1.2.1.16 ERRPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRPIDR4 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

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-22: EXT_ERRPIDR4 bit assignments



Table B-42: ERRPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	Size of the component. RAZ. Log ₂ of the number of 4KB pages from the start of the component to the end of the component ID registers. 0b0000 The component uses a single 4KB block.	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. 0b0100 Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B.	0b0100

Accessibility

Component	Offset	Instance	Range
RAS	0xFD0	ERRPIDR4	None

This interface is accessible as follows:

RO

B.1.2.1.17 ERRPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRPIDR0 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFE0

Access type

RO

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-23: EXT_ERRPIDR0 bit assignments

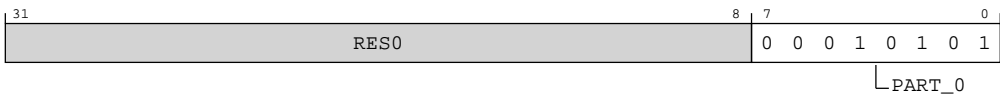


Table B-44: ERRPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b00010101 Cortex-R82 RAS component. Bits [7:0] of part number 0xD15.	0x15

Accessibility

Component	Offset	Instance	Range
RAS	0xFE0	ERRPIDR0	None

This interface is accessible as follows:

RO

B.1.2.1.18 ERRPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRPIDR1 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-24: EXT_ERRPIDR1 bit assignments

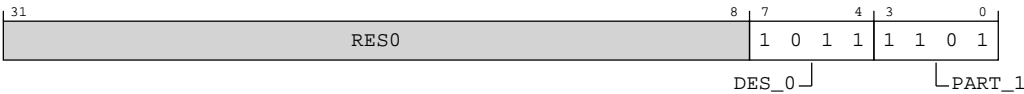


Table B-46: ERRPIDR1 bit descriptions

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. Bits [3:0] of JEP106 identification code 0x3B.	0b1011
[3:0]	PART_1	Part number bits [11:8]. 0b1101 Cortex-R82 RAS component. Bits [11:8] of part number 0xD15.	0b1101

Accessibility

Component	Offset	Instance	Range
RAS	0xFE4	ERRPIDR1	None

This interface is accessible as follows:

RO

B.1.2.1.19 ERRPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRPIDR2 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFE8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-25: EXT_ERRPIDR2 bit assignments

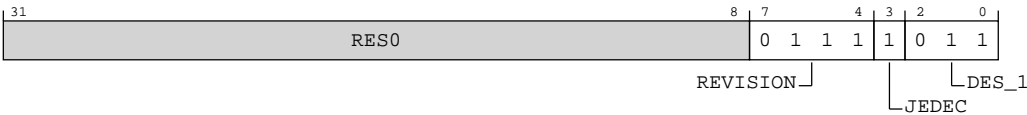


Table B-48: ERRPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111

Bits	Name	Description	Reset
[3]	JEDEC	RAO . Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Instance	Range
RAS	0xFE8	ERRPIDR2	None

This interface is accessible as follows:

RO

B.1.2.1.20 ERRPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For more information, see *About the Peripheral identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRPIDR3 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFEC

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-26: EXT_ERRPIDR3 bit assignments



Table B-50: ERRPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Part minor revision. Parts using ext-ERRPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
RAS	0xFEC	ERRPIDR3	None

This interface is accessible as follows:

RO

B.1.2.1.21 ERRCIDR0, Component Identification Register 0

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRCIDR0 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

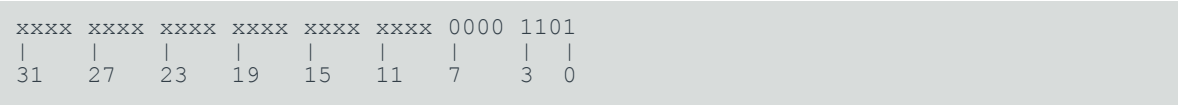
Register offset

0xFF0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-27: EXT_ERRCIDR0 bit assignments



Table B-52: ERRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble. 0b00001101 CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Instance	Range
RAS	0xFF0	ERRCIDR0	None

This interface is accessible as follows:

RO

B.1.2.1.22 ERRCIDR1, Component Identification Register 1

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRCIDR1 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFF4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-28: EXT_ERRCIDR1 bit assignments

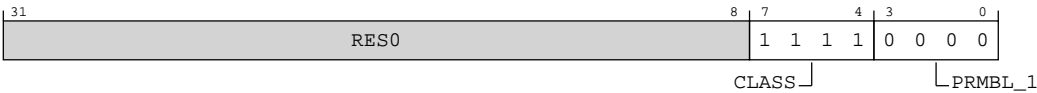


Table B-54: ERRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	CLASS	Component class. 0b1111 Generic peripheral with IMPLEMENTATION DEFINED register layout. Other values are defined by the CoreSight Architecture. This field reads as 0xF.	0b1111
[3:0]	PRMBL_1	Preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Instance	Range
RAS	0xFF4	ERRCIDR1	None

This interface is accessible as follows:

RO

B.1.2.1.23 ERRCIDR2, Component Identification Register 2

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRCIDR2 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-29: EXT_ERRCIDR2 bit assignments



Table B-56: ERRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble. 0b00000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Instance	Range
RAS	0xFF8	ERRCIDR2	None

This interface is accessible as follows:

RO

B.1.2.1.24 ERRCIDR3, Component Identification Register 3

Provides discovery information about the component.

For more information, see *About the Component Identification scheme* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

ERRCIDR3 is implemented only as part of a memory-mapped group of error records.

Attributes

Width

32

Component

RAS

Register offset

0xFFC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-30: EXT_ERRCIDR3 bit assignments

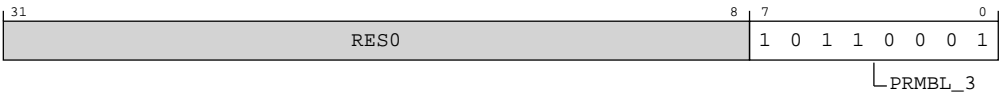


Table B-58: ERRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble. 0b10110001 CoreSight component identification preamble.	0xB1

Accessibility

Component	Offset	Instance	Range
RAS	0xFFC	ERRCIDR3	None

This interface is accessible as follows:

RO

B.1.2.2 External PPU register description

This section includes the register descriptions for all memory-mapped *Power Policy Unit* (PPU) registers that are accessed for each core in the Cortex®-R82 processor.

B.1.2.2.1 PPU_PWPR, Power Policy Register

This register enables software to program both power and operating mode policy. It also contains related settings including the enable for dynamic transitions and the lock enable.

This register does not reflect the current power mode value. The current power mode of the domain is reflected in the Power Status Register (ext-PPU_PWSR).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x000

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-31: EXT_PPU_PWPR bit assignments

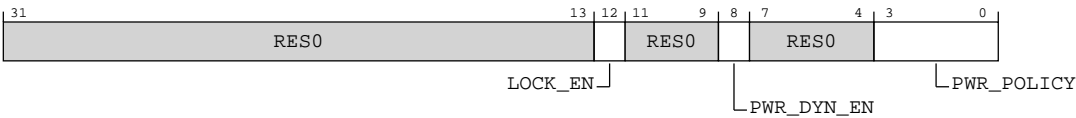


Table B-60: PPU_PWPR bit descriptions

Bits	Name	Description	Reset
[31:13]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[12]	LOCK_EN	Lock enable bit. 0b0 Lock feature disabled. 0b1 Lock feature enabled.	0b0
[11:9]	RES0	Reserved	RES0
[8]	PWR_DYN_EN	Power mode dynamic transition enable. 0b0 Dynamic transitions disabled for power modes. 0b1 Dynamic transitions enabled for power modes, allowing transitions to be initiated by changes on power mode DEVACTIVE inputs.	0b1
[7:4]	RES0	Reserved	RES0
[3:0]	PWR_POLICY	Power mode policy. When static power mode transitions are enabled, PWR_DYN_EN is set to 0b0, this is the target power mode for the PPU. When dynamic power mode transitions are enabled, PWR_DYN_EN is set to 0b1, this is the minimum power mode for the PPU. 0b0000 OFF. Logic off and RAM off. 0b0001 OFF_EMU. Emulated Off. Logic on with RAM on. This mode is used to emulate the functional condition of OFF without removing power. 0b0101 FULL_RET. Full Retention. Logic and RAM in retention. 0b0111 FUNC_RET. Functional Retention. Floating-point/Vector logic retained, rest of the core logic and RAM on, core is functional. 0b1000 ON. Logic on with RAM on, core is functional. 0b1001 WARM_RST. Warm Reset. Warm reset application with logic and RAM on. 0b1010 DBG_RECOV. Debug Recovery Reset. Warm reset application with logic and RAM on.	0b0000

Accessibility

Component	Offset	Range
PPU	0x000	None

This interface is accessible as follows:

RW

B.1.2.2.2 PPU_PMER, Power Mode Emulation Enable Register

This register allows software to enable entry into emulated modes.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

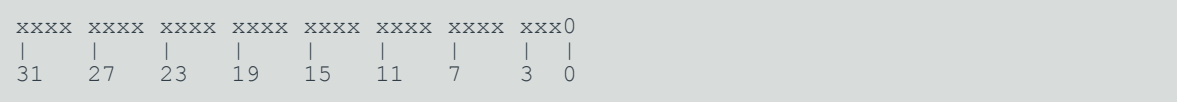
Register offset

0x004

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-32: EXT_PPU_PMER bit assignments

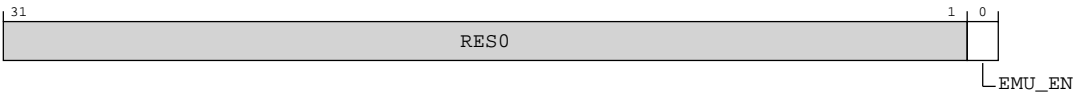


Table B-62: PPU_PMER bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	EMU_EN	Power mode emulation enable. 0b0 Power mode emulation disabled. 0b1 Power mode emulation enabled. Transitions to OFF and MEM_RET instead transition to OFF_EMU and MEM_RET_EMU.	0b0

Accessibility

Component	Offset	Range
PPU	0x004	None

This interface is accessible as follows:

RW

B.1.2.2.3 PPU_PWSR, Power Status Register

This read-only register contains status information for the power mode, operating mode, dynamic transitions, and lock feature.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x008

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxx0	xxx1	xxxx	0000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-33: EXT_PPU_PWSR bit assignments

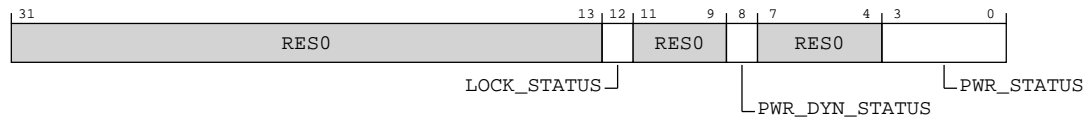


Table B-64: PPU_PWSR bit descriptions

Bits	Name	Description	Reset
[31:13]	RES0	Reserved	RES0
[12]	LOCK_STATUS	Lock status. 0b0 The PPU is not locked in the current mode. 0b1 The PPU is locked in the current mode.	0b0
[11:9]	RES0	Reserved	RES0
[8]	PWR_DYN_STATUS	Power mode dynamic transition status. There might be a delay in dynamic transitions becoming active or inactive if the PPU is transitioning when ext-PPU_PWPR.DYN_EN is programmed. 0b0 Dynamic transitions disabled for power modes. 0b1 Dynamic transitions enabled for power modes.	0b1
[7:4]	RES0	Reserved	RES0
[3:0]	PWR_STATUS	Power mode status. These bits reflect the current power mode of the PPU. 0b0000 OFF. Logic off and RAM off. 0b0001 OFF_EMU. Emulated Off. Logic on with RAM on. This mode is used to emulate the functional condition of OFF without removing power. 0b0101 FULL_RET. Full Retention. Logic and RAM in retention. 0b0111 FUNC_RET. Functional Retention. Floating-point/Vector logic retained, rest of the core logic and RAM on, core is functional. 0b1000 ON. Logic on with RAM on, core is functional. 0b1001 WARM_RST. Warm Reset. Warm reset application with logic and RAM on. 0b1010 DBG_RECOV. Debug Recovery Reset. Warm reset application with logic and RAM on.	0b0000

Accessibility

Component	Offset	Range
PPU	0x008	None

This interface is accessible as follows:

RO

B.1.2.2.4 PPU_DISR, Device Interface Input Current Status Register

This read-only register contains status reflecting the values of the device interface inputs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x010

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	x000	0000	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-34: EXT_PPU_DISR bit assignments

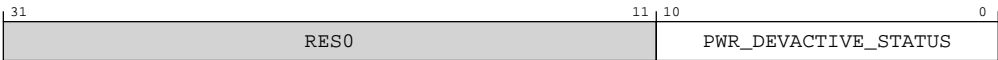


Table B-66: PPU_DISR bit descriptions

Bits	Name	Description	Reset
[31:11]	RES0	Reserved	RES0
[10:0]	PWR_DEVACTIVE_STATUS	<p>Status of the power mode DEVPACTIVE inputs.</p> <p>0b000000000000 Request for OFF.</p> <p>0000000001x Request for OFF_EMU.</p> <p>000001xxxxx Request for FULL_RET.</p> <p>0001xxxxxxx Request for FUNC_RET.</p> <p>001xxxxxxxx Request for ON.</p> <p>01xxxxxxxxx Request for WARM_RST.</p> <p>1xxxxxxxxxx Request for DBG_RECOV.</p>	0b000000000000

Accessibility

Component	Offset	Range
PPU	0x010	None

This interface is accessible as follows:

RO

B.1.2.2.5 PPU_MISR, Miscellaneous Input Current Status Register

This read-only register contains status reflecting the values of miscellaneous inputs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

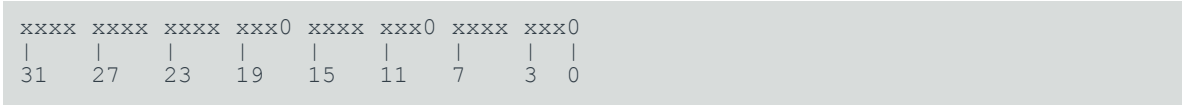
PPU

Register offset

0x014

Access type
RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-35: EXT_PPU_MISR bit assignments

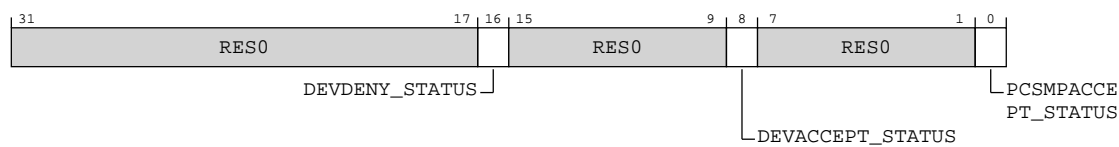


Table B-68: PPU_MISR bit descriptions

Bits	Name	Description	Reset
[31:17]	RES0	Reserved	RES0
[16]	DEVDPDENY_STATUS	Status of the device interface DEVDPDENY inputs. 0b0 DEVDPDENY deasserted. 0b1 DEVDPDENY asserted.	0b0
[15:9]	RES0	Reserved	RES0
[8]	DEVACCEPT_STATUS	Status of the device interface DEVPACCEPT inputs. 0b0 DEVPACCEPT deasserted. 0b1 DEVPACCEPT asserted.	0b0
[7:1]	RES0	Reserved	RES0
[0]	PCSMACCEPT_STATUS	Status of the PCSMPACCEPT inputs. 0b0 PCSMACCEPT deasserted. 0b1 PCSMACCEPT asserted.	0b0

Accessibility

Component	Offset	Range
PPU	0x014	None

This interface is accessible as follows:

RO

B.1.2.2.6 PPU_STSR, Stored Status Register

This register is reserved for P-channel PPUs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x018

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-36: EXT_PPU_STSR bit assignments

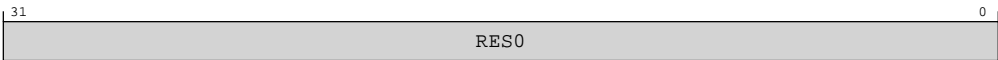


Table B-70: PPU_STSR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x018	None

This interface is accessible as follows:

RO

B.1.2.2.7 PPU_UNLK, Unlock Register

This register allows software to unlock the PPU from a locked power mode.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x01C

Access type

RAZW

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-37: EXT_PPU_UNLK bit assignments

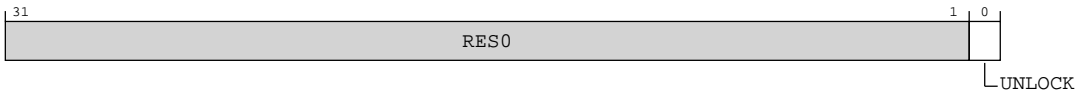


Table B-72: PPU_UNLK bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	UNLOCK	When 0b1 is written to this bit the PPU is unlocked from a locked power mode. A read always returns 0b0.	0b0

Accessibility

Component	Offset	Range
PPU	0x01C	None

This interface is accessible as follows:

RW

B.1.2.2.8 PPU_PWCR, Power Configuration Register

This register controls enabling and disabling of hardware control inputs to the PPU.



Before software programs the DEVREQEN bits it must configure the PPU for static transitions and ensure the requested power mode has been reached, this means that no further transitions can occur, otherwise behavior is UNPREDICTABLE.

The PWR_DEVACTIVEEN and OP_DEVACTIVEEN fields in this register control the ability of the DEVACTIVE inputs to initiate power mode transitions, but not the ability to generate input edge interrupt events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x020

Access type

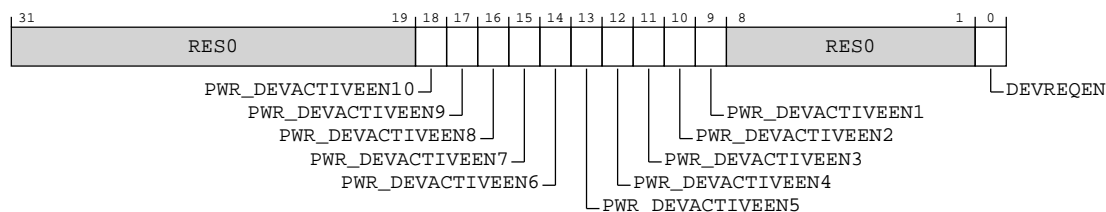
RW

Reset value

xxxx	xxxx	xxxx	x111	1111	111x	xxxx	xxx1
31	27	23	19	15	11	7	3 0

**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions**Figure B-38: EXT_PPU_PWCR bit assignments****Table B-74: PPU_PWCR bit descriptions**

Bits	Name	Description	Reset
[31:19]	RES0	Reserved	RES0
[18]	PWR_DEVACTIVEEN10	Enables the operating mode DEVPACTIVE[10] input. 0b0 DEVPACTIVE[10] input (DBG_RECOV) disabled. 0b1 DEVPACTIVE[10] input (DBG_RECOV) enabled.	0b1
[17]	PWR_DEVACTIVEEN9	Enables the operating mode DEVPACTIVE[9] input. 0b0 DEVPACTIVE[9] input (WARM_RST) disabled. 0b1 DEVPACTIVE[9] input (WARM_RST) enabled.	0b1
[16]	PWR_DEVACTIVEEN8	Enables the operating mode DEVPACTIVE[8] input. 0b0 DEVPACTIVE[8] input (ON) disabled. 0b1 DEVPACTIVE[8] input (ON) enabled.	0b1

Bits	Name	Description	Reset
[15]	PWR_DEVACTIVEEN7	Enables the operating mode DEVPACTIVE[7] input. 0b0 DEVPACTIVE[7] input (FUNC_RET) disabled. 0b1 DEVPACTIVE[7] input (FUNC_RET) enabled.	0b1
[14]	PWR_DEVACTIVEEN6	Enables the operating mode DEVPACTIVE[6] input. 0b0 DEVPACTIVE[6] input (unused) disabled. 0b1 DEVPACTIVE[6] input (unused) enabled.	0b1
[13]	PWR_DEVACTIVEEN5	Enables the operating mode DEVPACTIVE[5] input. 0b0 DEVPACTIVE[5] input (FULL_RET) disabled. 0b1 DEVPACTIVE[5] input (FULL_RET) enabled.	0b1
[12]	PWR_DEVACTIVEEN4	Enables the operating mode DEVPACTIVE[4] input. 0b0 DEVPACTIVE[4] input (unused) disabled. 0b1 DEVPACTIVE[4] input (unused) enabled.	0b1
[11]	PWR_DEVACTIVEEN3	Enables the operating mode DEVPACTIVE[3] input. 0b0 DEVPACTIVE[3] input (unused) disabled. 0b1 DEVPACTIVE[3] input (unused) enabled.	0b1
[10]	PWR_DEVACTIVEEN2	Enables the operating mode DEVPACTIVE[2] input. 0b0 DEVPACTIVE[2] input (unused) disabled. 0b1 DEVPACTIVE[2] input (unused) enabled.	0b1
[9]	PWR_DEVACTIVEEN1	Enables the operating mode DEVPACTIVE[1] input. 0b0 DEVPACTIVE[1] input (OFF_EMU) disabled. 0b1 DEVPACTIVE[1] input (OFF_EMU) enabled.	0b1
[8:1]	RES0	Reserved	RES0
[0]	DEVREQEN	Device interface handshake enable. 0b0 Device interface handshake disabled for transitions. 0b1 Device interface handshake enabled for transitions.	0b1

Accessibility

Component	Offset	Range
PPU	0x020	None

This interface is accessible as follows:

RW

B.1.2.2.9 PPU_PTCR, Power Mode Transition Register

This register contains settings which affect the behaviour of certain power mode transitions.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x024

Access type

RW

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx01
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-39: EXT_PPU_PTCR bit assignments

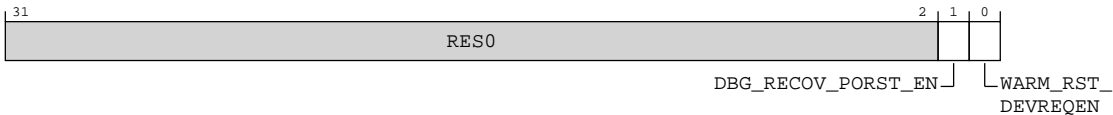


Table B-76: PPU_PTCR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	DBG_RECOV_PORST_EN	<p>Power-on reset behavior in DBG_RECOV.</p> <p>This bit should not be modified when the PPU is in DBG_RECOV. If it is modified, it will not have any effect on the reset output.</p> <p>This bit should not be modified when the PPU is in transition. If it is modified, the reset output will depend on either the old or the new value of the bit.</p> <p>0b0</p> <p>DEVPORESETn is not asserted when in DBG_RECOV.</p> <p>0b1</p> <p>DEVPORESETn is asserted when in DBG_RECOV.</p>	0b0
[0]	WARM_RST_DEVREQEN	<p>Device interface handshake behavior.</p> <p>This bit should not be modified when the PPU is in WARM_RST, or if the PPU is performing a transition, otherwise PPU behavior is UNPREDICTABLE.</p> <p>0b0</p> <p>The PPU does not perform a device interface handshake when transitioning between ON and WARM_RST.</p> <p>0b1</p> <p>The PPU performs a device interface handshake when transitioning between ON and WARM_RST.</p>	0b1

Accessibility

Component	Offset	Range
PPU	0x024	None

This interface is accessible as follows:

RW

B.1.2.2.10 PPU_IMR, Interrupt Mask Register

This register controls the events that assert the interrupt output. Additional event masking controls are in the Additional Interrupt Mask Register (ext-PPU_AIMR), Input Edge Sensitivity Register (ext-PPU_IESR), and the Operating Mode Active Edge Sensitivity Register (ext-PPU_OPSR).

When an interrupt event is masked an occurrence of the event does not set the corresponding bit in the interrupt status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x030

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-40: EXT_PPU_IMR bit assignments

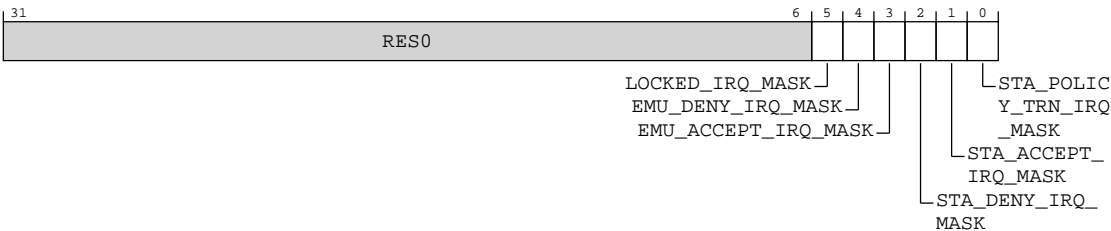


Table B-78: PPU_IMR bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	LOCKED_IRQ_MASK	Locked event mask 0b0 Locked event enabled. 0b1 Locked event masked.	0b1

Bits	Name	Description	Reset
[4]	EMU_DENY_IRQ_MASK	Emulation transition denial event mask 0b0 Emulation transition denial event enabled. 0b1 Emulation transition denial event masked.	0b1
[3]	EMU_ACCEPT_IRQ_MASK	Emulation transition acceptance event mask 0b0 Emulation transition acceptance event enabled. 0b1 Emulation transition acceptance event masked.	0b1
[2]	STA_DENY_IRQ_MASK	Static transition denial event mask 0b0 Static transition denial event enabled. 0b1 Static transition denial event masked.	0b0
[1]	STA_ACCEPT_IRQ_MASK	Static transition acceptance event mask 0b0 Static transition acceptance event enabled. 0b1 Static transition acceptance event masked.	0b1
[0]	STA_POLICY_TRN_IRQ_MASK	Static full policy transition completion event mask 0b0 Static full policy transition completion event enabled. 0b1 Static full policy transition completion event masked.	0b0

Accessibility

Component	Offset	Range
PPU	0x030	None

This interface is accessible as follows:

RW

B.1.2.2.11 PPU_AIMR, Additional Interrupt Mask Register

This register controls the events that assert the interrupt output. Additional event masking controls are in the Interrupt Mask Register (ext-PPU_IMR), Input Edge Sensitivity Register (ext-PPU_IESR), and the Operating Mode Active Edge Sensitivity Register (ext-PPU_OPSR).

When an interrupt event is masked an occurrence of the event does not set the corresponding bit in the interrupt status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

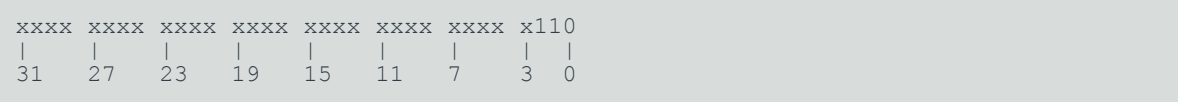
Register offset

0x034

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-41: EXT_PPU_AIMR bit assignments

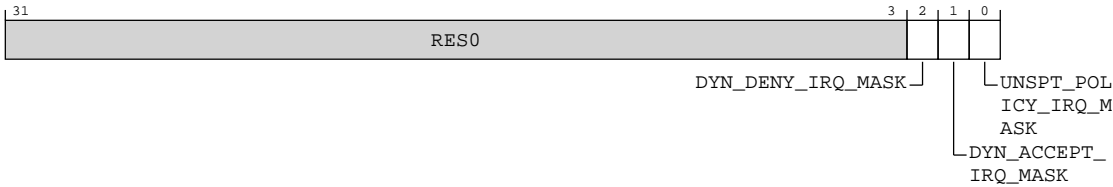


Table B-80: PPU_AIMR bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0
[2]	DYN_DENY_IRQ_MASK	Dynamic transition denial event mask 0b0 Dynamic transition denial event enabled. 0b1 Dynamic transition denial event masked.	0b1

Bits	Name	Description	Reset
[1]	DYN_ACCEPT_IRQ_MASK	Dynamic transition acceptance event mask 0b0 Dynamic transition acceptance event enabled. 0b1 Dynamic transition acceptance event masked.	0b1
[0]	UNSPT_POLICY_IRQ_MASK	Unsupported policy event mask 0b0 Unsupported policy event enabled. 0b1 Unsupported policy event masked.	0b0

Accessibility

Component	Offset	Range
PPU	0x034	None

This interface is accessible as follows:

RW

B.1.2.2.12 PPU_ISR, Interrupt Status Register

This register contains information about events causing the assertion of the interrupt output. It is also used to clear interrupt events.

A bit set to 0b1 indicates the event asserted the interrupt output. Multiple events can be active at the same time. When an interrupt event is masked an occurrence of that event does not set the status bit.

A write of 0b1 to an event bit clears that event. A write of 0b0 to a bit has no effect. The interrupt output stays HIGH until all status bits in the Interrupt Status Register (PPU_ISR) and the Additional Interrupt Status Register (ext-PPU_AISR) are 0b0.

When the OTHER_IRQ bit is set, this indicates an event from the Additional Interrupt Status Register (PPU_AISR) has caused the interrupt output to be asserted. This bit cannot be cleared by writing to this register. It must be cleared by writing to the active event in the Additional Interrupt Status Register (ext-PPU_AISR).

Configurations

This register is available in all configurations.

Attributes

Width

32

Bits	Name	Description	Reset
[16]	PWR_ACTIVE_EDGE_IRQ8	Indicates if power mode DEVPACTIVE[8] input caused the input edge event. 0b0 DEVPACTIVE[8] input (ON) did not assert the interrupt output. 0b1 DEVPACTIVE[8] input (ON) asserted the interrupt output.	0b0
[15]	PWR_ACTIVE_EDGE_IRQ7	Indicates if power mode DEVPACTIVE[7] input caused the input edge event. 0b0 DEVPACTIVE[7] input (FUNC_RET) did not assert the interrupt output. 0b1 DEVPACTIVE[7] input (FUNC_RET) asserted the interrupt output.	0b0
[14]	RES0	Reserved	RES0
[13]	PWR_ACTIVE_EDGE_IRQ5	Indicates if power mode DEVPACTIVE[5] input caused the input edge event. 0b0 DEVPACTIVE[5] input (FULL_RET) did not assert the interrupt output. 0b1 DEVPACTIVE[5] input (FULL_RET) asserted the interrupt output.	0b0
[12:10]	RES0	Reserved	RES0
[9]	PWR_ACTIVE_EDGE_IRQ1	Indicates if power mode DEVPACTIVE[1] input caused the input edge event. 0b0 DEVPACTIVE[1] input (OFF_EMU) did not assert the interrupt output. 0b1 DEVPACTIVE[1] input (OFF_EMU) asserted the interrupt output.	0b0
[8]	RES0	Reserved	RES0
[7]	OTHER_IRQ	Indicates there is an interrupt event pending in the Additional Interrupt Status Register (ext-PPU_AISR). 0b0 No interrupt pending in ext-PPU_AISR. 0b1 Interrupt pending in ext-PPU_AISR.	0b0
[6]	RES0	Reserved	RES0
[5]	LOCKED_IRQ	Locked event status. 0b0 No locked event. 0b1 A locked event asserted the interrupt output.	0b0
[4]	EMU_DENY_IRQ	Emulated transition denial event status. 0b0 No emulated transition denial event. 0b1 An emulated transition denial event asserted the interrupt output.	0b0

Bits	Name	Description	Reset
[3]	EMU_ACCEPT_IRQ	Emulated transition acceptance event status. 0b0 No emulated transition acceptance event. 0b1 An emulated transition acceptance event asserted the interrupt output.	0b0
[2]	STA_DENY_IRQ	Static transition denial event status. 0b0 No static transition denial event. 0b1 An static transition denial event asserted the interrupt output.	0b0
[1]	STA_ACCEPT_IRQ	Static transition acceptance event status. 0b0 No static transition acceptance event. 0b1 An static transition acceptance event asserted the interrupt output.	0b0
[0]	STA_POLICY_TRN_IRQ	Static full policy transition completion event status. 0b0 No static full policy transition completion event. 0b1 An static full policy transition completion event asserted the interrupt output.	0b0

Accessibility

Component	Offset	Range
PPU	0x038	None

This interface is accessible as follows:

RW

B.1.2.2.13 PPU_AISR, Additional Interrupt Status Register

This register contains information about events causing the assertion of the interrupt output. It is also used to clear interrupt events.

A bit set to 0b1 indicates the event asserted the interrupt output. Multiple events can be active at the same time. When an interrupt event is masked an occurrence of that event does not set the status bit.

A write of 0b1 to an event bit clears that event. A write of 0b0 has no effect. The interrupt output stays HIGH until all status bits in the Interrupt Status Register (ext-PPU_ISR) and the Additional Interrupt Status Register (PPU_AISR) are set to 0b0.

When an interrupt status is set to 0b1 in this register it sets the OTHER_IRQ bit in the Interrupt Status Register (ext-PPU_ISR). Status bits in this register are only cleared by writing to this register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x03C

Access type

RW1C

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-43: EXT_PPU_AISR bit assignments

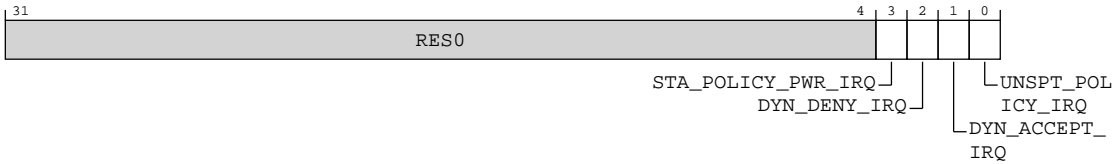


Table B-84: PPU_AISR bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3]	STA_POLICY_PWR_IRQ	Static power policy transition completion event status 0b0 No static power policy transition completion event. 0b1 A static power policy transition completion event asserted the interrupt output.	0b0

Bits	Name	Description	Reset
[2]	DYN_DENY_IRQ	Dynamic transition denial event status 0b0 No dynamic transition denial event. 0b1 A dynamic transition denial event asserted the interrupt output.	0b0
[1]	DYN_ACCEPT_IRQ	Dynamic transition acceptance event status 0b0 No dynamic transition acceptance event. 0b1 A dynamic transition acceptance event asserted the interrupt output.	0b0
[0]	UNSPT_POLICY_IRQ	Unsupported policy event status 0b0 No unsupported policy event. 0b1 An unsupported policy event asserted the interrupt output.	0b0

Accessibility

Component	Offset	Range
PPU	0x03C	None

This interface is accessible as follows:

RW

B.1.2.2.14 PPU_IESR, Input Edge Sensitivity Register

This register configures the transitions on the power mode DEVPACTIVE inputs that generate an Input Edge interrupt event.

When an event is masked an occurrence of the event does not set the corresponding bit in the interrupt status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x040

Access type
RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-44: EXT_PPU_IESR bit assignments

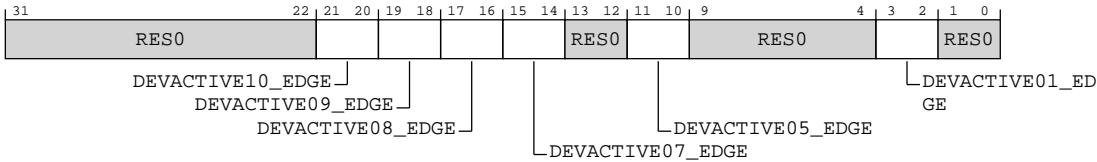


Table B-86: PPU_IESR bit descriptions

Bits	Name	Description	Reset
[31:22]	RES0	Reserved	RES0
[21:20]	DEVACTIVE10_EDGE	Configures the transitions on the DEVPACTIVE[10] input (DBG_RECOV) that generate an Input Edge interrupt event. 0b00 Event maksed. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00

Bits	Name	Description	Reset
[19:18]	DEVACTIVE09_EDGE	Configures the transitions on the DEVPACTIVE[9] input (WARM_RST) that generate an Input Edge interrupt event. 0b00 Event masked. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00
[17:16]	DEVACTIVE08_EDGE	Configures the transitions on the DEVPACTIVE[8] input (ON) that generate an Input Edge interrupt event. 0b00 Event masked. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00
[15:14]	DEVACTIVE07_EDGE	Configures the transitions on the DEVPACTIVE[7] input (FUNC_RET) that generate an Input Edge interrupt event. 0b00 Event masked. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00
[13:12]	RES0	Reserved	RES0
[11:10]	DEVACTIVE05_EDGE	Configures the transitions on the DEVPACTIVE[5] input (FULL_RET) that generate an Input Edge interrupt event. 0b00 Event masked. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00
[9:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:2]	DEVACTIVE01_EDGE	Configures the transitions on the DEVPACTIVE[1] input (OFF_EMU) that generate an Input Edge interrupt event. 0b00 Event maksed. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00
[1:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x040	None

This interface is accessible as follows:

RW

B.1.2.2.15 PPU_OPSR, Operating Mode Active Edge Sensitivity Register

This register configures the transitions on the operating mode DEVPACTIVE inputs that generate an Input Edge interrupt event.

When an event is masked an occurrence of the event does not set the corresponding bit in the interrupt status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x044

Access type

RW

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
------	------	------	------	------	------	------	------



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-45: EXT_PPU_OPSR bit assignments

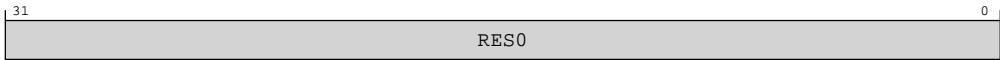


Table B-88: PPU_OPSR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x044	None

This interface is accessible as follows:

RW

B.1.2.2.16 PPU_FUNRR, Functional Retention RAM Configuration Register

This register is reserved.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

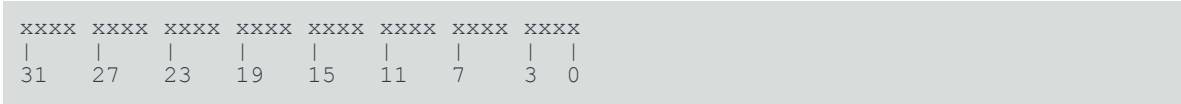
Register offset

0x050

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-46: EXT_PPU_FUNRR bit assignments

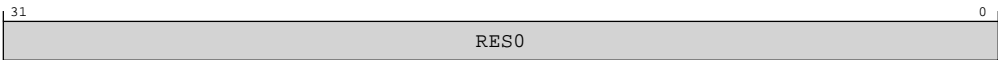


Table B-90: PPU_FUNRR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x050	None

This interface is accessible as follows:

RW

B.1.2.2.17 PPU_FULRR, Full Retention RAM Configuration Register

This register is reserved (high bits of PCSMPSTATE outputs are not configurable by software).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x054

Access type
RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-47: EXT_PPU_FULRR bit assignments



Table B-92: PPU_FULRR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x054	None

This interface is accessible as follows:

RW

B.1.2.2.18 PPU_MEMRR, Memory Retention RAM Configuration Register

This register is reserved (high bits of PCSMPSTATE outputs are not configurable by software).

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
PPU

Register offset

0x058

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-48: EXT_PPU_MEMRR bit assignments

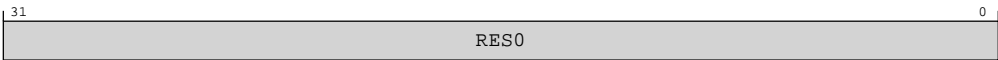


Table B-94: PPU_MEMRR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x058	None

This interface is accessible as follows:

RW

B.1.2.2.19 PPU_EDTR0, Power Mode Entry Delay Register 0

This register is reserved (power mode entry delays not supported).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x160

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-49: EXT_PPU_EDTR0 bit assignments

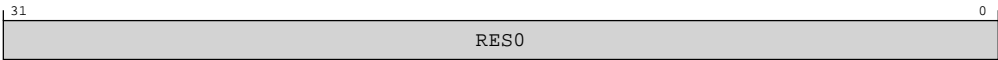


Table B-96: PPU_EDTR0 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x160	None

This interface is accessible as follows:

RW

B.1.2.2.20 PPU_EDTR1, Power Mode Entry Delay Register 1

This register is reserved (power mode entry delays not supported).

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
PPU

Register offset
0x164

Access type
RW

Reset value

xxxx

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-50: EXT_PPU_EDTR1 bit assignments

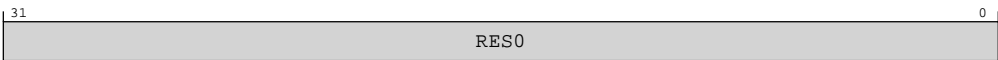


Table B-98: PPU_EDTR1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0x164	None

This interface is accessible as follows:

RW

B.1.2.2.21 PPU_DCDR0, Device Control Delay Configuration Register 0

This register is used to program device control delay parameters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x170

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-51: EXT_PPU_DCDR0 bit assignments

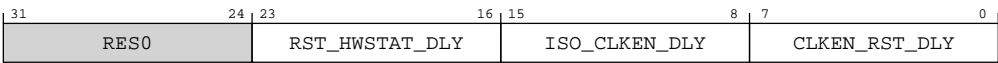


Table B-100: PPU_DCDR0 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:16]	RST_HWSTAT_DLY	Delay from reset de-assertion to HWSTAT update.	0x00
[15:8]	ISO_CLKEN_DLY	Delay from isolation enable de-assertion to clock enable assertion.	0x00
[7:0]	CLKEN_RST_DLY	Delay from clock enable assertion to reset de-assertion.	0x00

Accessibility

Component	Offset	Range
PPU	0x170	None

This interface is accessible as follows:

RW

B.1.2.2.22 PPU_DCDR1, Device Control Delay Configuration Register 1

This register is used to program device control delay parameters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0x174

Access type

RW

Reset value

xxxx	xxxx	xxxx	xxxx	0000	0000	0000	0000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-52: EXT_PPU_DCDR1 bit assignments

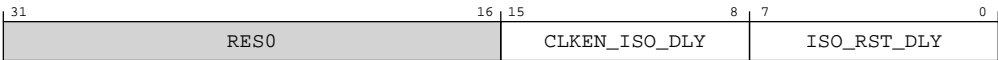


Table B-102: PPU_DCDR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:8]	CLKEN_ISO_DLY	Delay from clock enable de-assertion to isolation enable assertion.	0x00
[7:0]	ISO_RST_DLY	Delay from isolation enable assertion to reset assertion.	0x00

Accessibility

Component	Offset	Range
PPU	0x174	None

This interface is accessible as follows:

RW

B.1.2.2.23 PPU_IDR0, PPU Identification Register 0

This read-only register contains information on the type and number of channels on the device interface and power and operating modes supported.

Additional information on optional features can be found in the PPU Identification Register 1 (ext-PPU_IDR1).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFB0

Access type

RO

Reset value

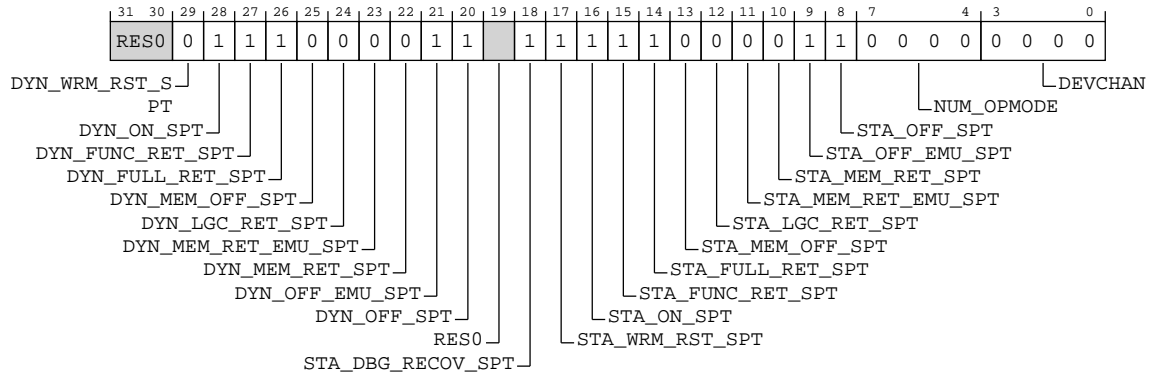
```

xx01 1100 0011 x111 1100 0011 0000 0000
|    |    |    |    |    |    |    |
31   27   23   19   15   11   7     3     0

```

**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions**Figure B-53: EXT_PPU_IDR0 bit assignments****Table B-104: PPU_IDR0 bit descriptions**

Bits	Name	Description	Reset
[31:30]	RES0	Reserved	RES0
[29]	DYN_WRM_RST_SPT	Dynamic WARM_RST support. 0b0 Dynamic WARM_RST not supported.	0b0
[28]	DYN_ON_SPT	Dynamic ON support. 0b1 Dynamic ON supported.	0b1
[27]	DYN_FUNC_RET_SPT	Dynamic DYN_FUNC_RET_SPT support. 0b1 Dynamic DYN_FUNC_RET_SPT supported.	0b1
[26]	DYN_FULL_RET_SPT	Dynamic DYN_FULL_RET_SPT support. 0b1 Dynamic DYN_FULL_RET_SPT supported.	0b1
[25]	DYN_MEM_OFF_SPT	Dynamic MEM_OFF support. 0b0 Dynamic MEM_OFF not supported.	0b0
[24]	DYN_LGC_RET_SPT	Dynamic LOGIC_RET support. 0b0 Dynamic LOGIC_RET not supported.	0b0

Bits	Name	Description	Reset
[23]	DYN_MEM_RET_EMU_SPT	Dynamic DYN_MEM_RET_EMU_SPT support. 0b0 Dynamic DYN_MEM_RET_EMU_SPT not supported.	0b0
[22]	DYN_MEM_RET_SPT	Dynamic DYN_MEM_RET_SPT support. 0b0 Dynamic DYN_MEM_RET_SPT not supported.	0b0
[21]	DYN_OFF_EMU_SPT	Dynamic OFF_EMU support. 0b1 Dynamic OFF_EMU supported.	0b1
[20]	DYN_OFF_SPT	Dynamic OFF support. 0b1 Dynamic OFF supported.	0b1
[19]	RES0	Reserved	RES0
[18]	STA_DBG_RECOV_SPT	DBG_RECOV support. 0b1 DBG_RECOV supported.	0b1
[17]	STA_WRM_RST_SPT	WRM_RST support. 0b1 WRM_RST supported.	0b1
[16]	STA_ON_SPT	ON support. 0b1 ON supported.	0b1
[15]	STA_FUNC_RET_SPT	FUNC_RET support. 0b1 FUNC_RET supported.	0b1
[14]	STA_FULL_RET_SPT	FULL_RET support. 0b1 FULL_RET supported.	0b1
[13]	STA_MEM_OFF_SPT	MEM_OFF support. 0b0 MEM_OFF not supported.	0b0
[12]	STA_LGC_RET_SPT	LOGIC_RET support. 0b0 LOGIC_RET not supported.	0b0
[11]	STA_MEM_RET_EMU_SPT	MEM_RET_EMU support. 0b0 MEM_RET_EMU not supported.	0b0
[10]	STA_MEM_RET_SPT	MEM_RET support. 0b0 MEM_RET not supported.	0b0

Bits	Name	Description	Reset
[9]	STA_OFF_EMU_SPT	OFF_EMU support. 0b1 OFF_EMU supported.	0b1
[8]	STA_OFF_SPT	OFF support. 0b1 OFF supported.	0b1
[7:4]	NUM_OPMODE	No. of operating modes supported, minus 1. 0b0000 1 operating mode supported.	0b0000
[3:0]	DEVCHAN	No. of Device Interface Channels. 0b0000 0 (P-channel PPU).	0b0000

Accessibility

Component	Offset	Range
PPU	0xFB0	None

This interface is accessible as follows:

RO

B.1.2.2.24 PPU_IDR1, PPU Identification Register 1

This read-only register contains information on the optional features and configurations that are supported by this PPU.

Additional information on optional features can be found in the PPU Identification Register 0 (ext-PPU_IDR0).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFB4

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxx0	xxx0	x000	x110
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-54: EXT_PPU_IDR1 bit assignments

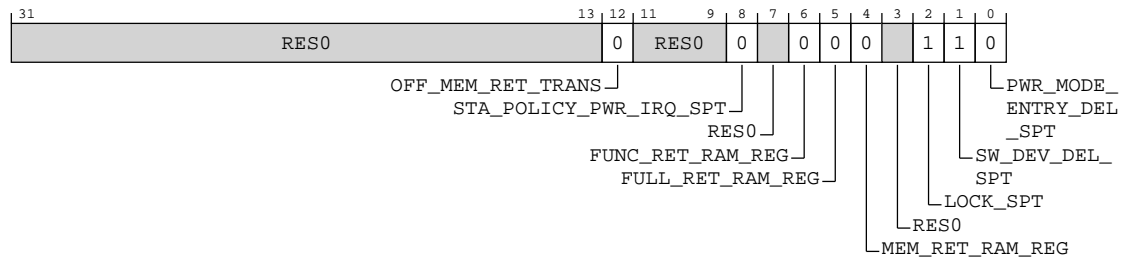


Table B-106: PPU_IDR1 bit descriptions

Bits	Name	Description	Reset
[31:13]	RES0	Reserved	RES0
[12]	OFF_MEM_RET_TRANS	OFF to MEM_RET direct transition. Indicates if direct transitions from OFF to MEM_RET and from OFF_EMU to MEM_RET_EMU are supported. 0b0 OFF to MEM_RET direct transition not supported.	0b0
[11:9]	RES0	Reserved	RES0
[8]	STA_POLICY_PWR_IRQ_SPT	Power policy transition completion event status. 0b0 Power policy transition completion events not supported.	0b0
[7]	RES0	Reserved	RES0
[6]	FUNC_RET_RAM_REG	Indicates if the ext-PPU_FUNRR register is present or reserved. 0b0 ext-PPU_FUNRR is reserved.	0b0
[5]	FULL_RET_RAM_REG	Indicates if the ext-PPU_FULRR register is present or reserved. 0b0 ext-PPU_FULRR is not present.	0b0
[4]	MEM_RET_RAM_REG	Indicates if the ext-PPU_MEMRR register is present or reserved. 0b0 ext-PPU_MEMRR is reserved.	0b0
[3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2]	LOCK_SPT	Indicates if the lock and the lock interrupt event are supported. 0b1 Lock and the lock interrupt event are supported.	0b1
[1]	SW_DEV_DEL_SPT	Software device delay control configuration support. 0b1 Software device delay control configuration supported.	0b1
[0]	PWR_MODE_ENTRY_DEL_SPT	Power mode entry delay support. 0b0 Power mode entry delay not supported.	0b0

Accessibility

Component	Offset	Range
PPU	0xFB4	None

This interface is accessible as follows:

RO

B.1.2.2.25 PPU_IIDR, Implementation Identification Register

This register provides information about the implementer and implementation of the PPU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFC8

Access type

RO

Reset value

0000 1011 0110 0001 0001 0100 0011 1011

Bit descriptions

Figure B-55: EXT_PPU_IIDR bit assignments

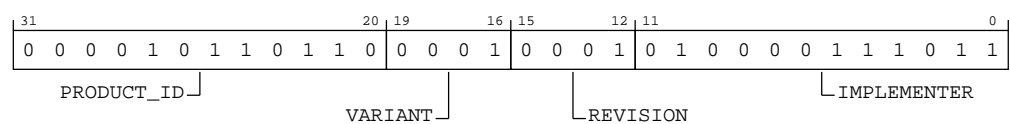


Table B-108: PPU_IIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	PRODUCT_ID	Value identifying the PPU part. 0b000010110110 PCK-600 PPU.	0x0B6
[19:16]	VARIANT	Value used to distinguish product variants, or major revisions of the product. 0b0001 Product variant r0p5.	0b0001
[15:12]	REVISION	Value used to distinguish minor revisions of the product. 0b0001 No ECO fixes.	0b0001
[11:0]	IMPLEMENTER	Implementer identification. 0b010000111011 Arm Limited.	0x43B

Accessibility

Component	Offset	Range
PPU	0xFC8	None

This interface is accessible as follows:

RO

B.1.2.2.26 PPU_AIDR, Architecture Identification Register

This register identifies the PPU architecture revision.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFCC

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	0001
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-56: EXT_PPU_AIDR bit assignments

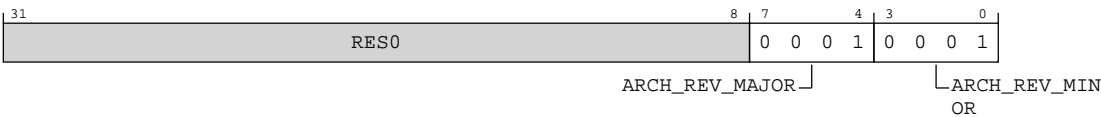


Table B-110: PPU_AIDR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	ARCH_REV_MAJOR	PPU architecture major revision. 0b0001 PPU architecture major revision 1.	0b0001
[3:0]	ARCH_REV_MINOR	PPU architecture minor revision. 0b0001 PPU architecture minor revision 1.	0b0001

Accessibility

Component	Offset	Range
PPU	0xFCC	None

This interface is accessible as follows:

RO

B.1.2.2.27 PPU_PIDR4, PPU Peripheral Identification Register 4

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFD0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-57: EXT_PPU_PIDR4 bit assignments

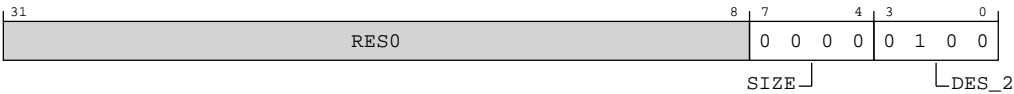


Table B-112: PPU_PIDR4 bit descriptions

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
PPU	0xFD0	None

This interface is accessible as follows:

RO

B.1.2.2.28 PPU_PIDR5, PPU Peripheral Identification Register 5

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFD4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-58: EXT_PPU_PIDR5 bit assignments



Table B-114: PPU_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0xFD4	None

This interface is accessible as follows:

RO

B.1.2.2.29 PPU_PIDR6, PPU Peripheral Identification Register 6

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFD8

Access type

RO

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-59: EXT_PPU_PIDR6 bit assignments

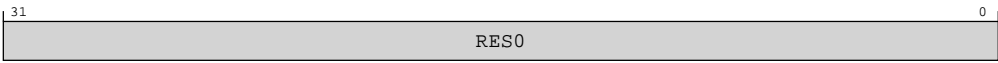


Table B-116: PPU_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0xFD8	None

This interface is accessible as follows:

RO

B.1.2.2.30 PPU_PIDR7, PPU Peripheral Identification Register 7

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

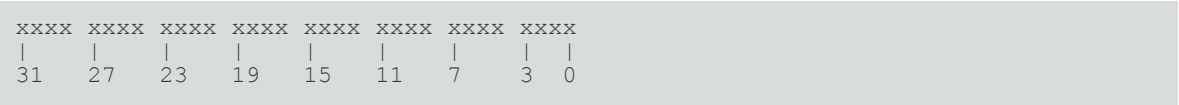
Register offset

0xFDC

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-60: EXT_PPU_PIDR7 bit assignments

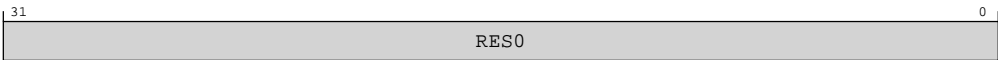


Table B-118: PPU_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
PPU	0xFDC	None

This interface is accessible as follows:

RO

B.1.2.2.31 PPU_PIDR0, PPU Peripheral Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFE0

Access type

RO

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 1011 0110
```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-61: EXT_PPU_PIDR0 bit assignments



Table B-120: PPU_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b10110110 PCK-600 PPU. Bits [7:0] of part number 0x0B6.	0xB6

Accessibility

Component	Offset	Range
PPU	0xFE0	None

This interface is accessible as follows:

RO

B.1.2.2.32 PPU_PIDR1, PPU Peripheral Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFE4

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1011	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-62: EXT_PPU_PIDR1 bit assignments



Table B-122: PPU_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]. 0b0000 PCK-600 PPU. Bits [11:8] of part number 0x0B6.	0b0000

Accessibility

Component	Offset	Range
PPU	0xFE4	None

This interface is accessible as follows:

RO

B.1.2.2.33 PPU_PIDR2, PPU Peripheral Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFE8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-63: EXT_PPU_PIDR2 bit assignments

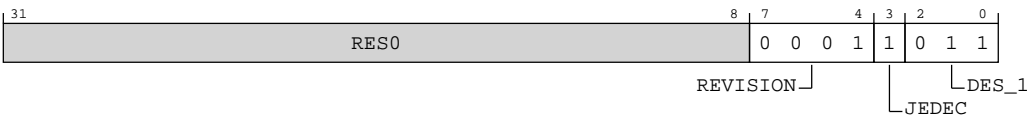


Table B-124: PPU_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0001 Revision r0p5.	0b0001

Bits	Name	Description	Reset
[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
PPU	0xFE8	None

This interface is accessible as follows:

RO

B.1.2.2.34 PPU_PIDR3, PPU Peripheral Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

Register offset

0xFE8

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-64: EXT_PPU_PIDR3 bit assignments

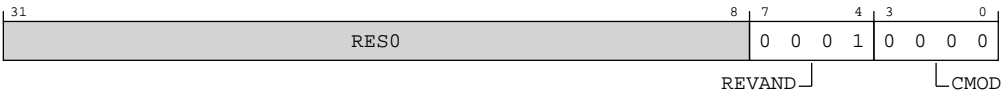


Table B-126: PPU_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Minor errata fixes. 0b0001 No ECO fixes.	0b0001
[3:0]	CMOD	Customer Modified. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
PPU	0xFEC	None

This interface is accessible as follows:

RO

B.1.2.2.35 PPU_CIDR0, PPU Component Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

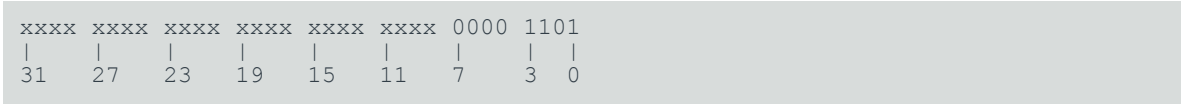
Register offset

0xFF0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-65: EXT_PPU_CIDR0 bit assignments



Table B-128: PPU_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
PPU	0xFF0	None

This interface is accessible as follows:

RO

B.1.2.2.36 PPU_CIDR1, PPU Component Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

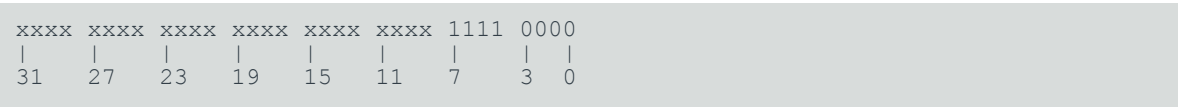
Register offset

0xFF4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-66: EXT_PPU_CIDR1 bit assignments



Table B-130: PPU_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. 0b1111 CoreLink component.	0b1111
[3:0]	PRMBL_1	CoreSight component identification preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
PPU	0xFF4	None

This interface is accessible as follows:

RO

B.1.2.2.37 PPU_CIDR2, PPU Component Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

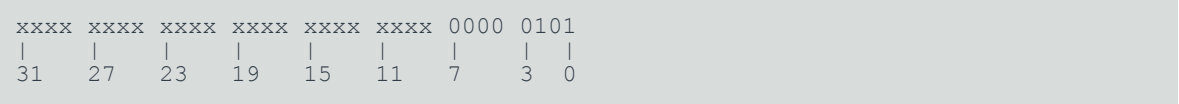
Register offset

0xFF8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-67: EXT_PPU_CIDR2 bit assignments

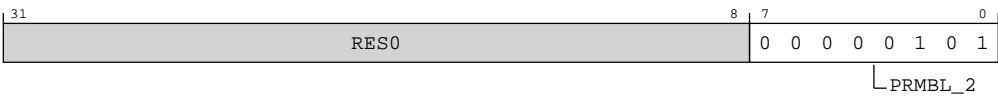


Table B-132: PPU_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	CoreSight component identification preamble. 0b000000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
PPU	0xFF8	None

This interface is accessible as follows:

RO

B.1.2.2.38 PPU_CIDR3, PPU Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

PPU

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-68: EXT_PPU_CIDR3 bit assignments

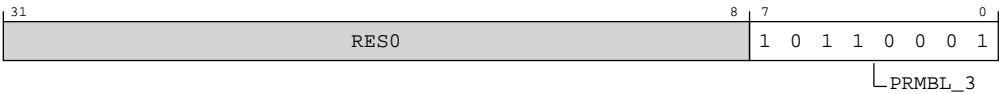


Table B-134: PPU_CIDR3 bit descriptions

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
PPU	0xFFC	None

This interface is accessible as follows:

RO

B.1.2.3 External CLUSTERPPU register description

This section includes the register descriptions for all memory-mapped CLUSTERPPU registers that are accessed for the cluster.

B.1.2.3.1 CLUSTERPPU_PWPR, Power Policy Register

This register enables software to program both power and operating mode policy. It also contains related settings including the enable for dynamic transitions and the lock enable.

This register does not reflect the current power mode value. The current power mode of the domain is reflected in the Power Status Register (ext-CLUSTERPPU_PWSR).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x000

Access type

RW

Reset value

xxxx xxx0 xxxx xxxx xxx0 xxx1 xxxx 0000



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-69: EXT_CLUSTERPPU_PWPR bit assignments

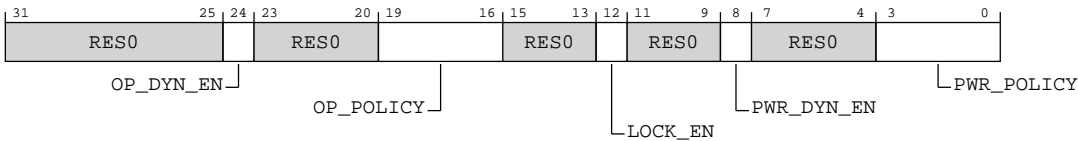


Table B-136: CLUSTERPPU_PWPR bit descriptions

Bits	Name	Description	Reset
[31:25]	RES0	Reserved	RES0
[24]	OP_DYN_EN	Operating mode dynamic transition enable. 0b0 Dynamic transitions disabled for operating modes. 0b1 Dynamic transitions enabled for operating modes, allowing transitions to be initiated by changes on operating mode DEVACTIVE inputs.	0b0
[23:20]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[19:16]	OP_POLICY	<p>When L2_SLICES == 2</p> <p>Operating mode policy.</p> <p>When static operating mode transitions are enabled, OP_DYN_EN is set to 0b0, then this is the target operating mode for the PPU.</p> <p>When dynamic operating mode transitions are enabled, OP_DYN_EN is set to 0b1, then this is the minimum operating mode for the PPU.</p> <p>0b0000</p> <p>OPMODE_00: The L2 Cache is not operational.</p> <p>0b0001</p> <p>OPMODE_01: Half of the L2 Cache is operational.</p> <p>0b0010</p> <p>OPMODE_02: The whole L2 Cache is operational.</p> <p>Otherwise</p> <p>Operating mode policy.</p> <p>When static operating mode transitions are enabled, OP_DYN_EN is set to 0b0, then this is the target operating mode for the PPU.</p> <p>When dynamic operating mode transitions are enabled, OP_DYN_EN is set to 0b1, then this is the minimum operating mode for the PPU.</p> <p>0b0000</p> <p>OPMODE_00: The L2 Cache is not operational.</p> <p>0b0010</p> <p>OPMODE_02: The whole L2 Cache is operational.</p>	xxxx
[15:13]	RES0	Reserved	RES0
[12]	LOCK_EN	<p>Lock enable bit.</p> <p>0b0</p> <p>Lock feature disabled.</p> <p>0b1</p> <p>Lock feature enabled.</p>	0b0
[11:9]	RES0	Reserved	RES0
[8]	PWR_DYN_EN	<p>Power mode dynamic transition enable.</p> <p>0b0</p> <p>Dynamic transitions disabled for power modes.</p> <p>0b1</p> <p>Dynamic transitions enabled for power modes, allowing transitions to be initiated by changes on power mode DEACTIVE inputs.</p>	0b1
[7:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:0]	PWR_POLICY	<p>Power mode policy.</p> <p>When static power mode transitions are enabled, PWR_DYN_EN is set to 0b0, this is the target power mode for the CLUSTERPPU.</p> <p>When dynamic power mode transitions are enabled, PWR_DYN_EN is set to 0b1, this is the minimum power mode for the CLUSTERPPU.</p> <p>0b0000 OFF. Logic off and RAM off.</p> <p>0b0001 OFF_EMU. Emulated Off. Logic on with RAM on. This mode is used to emulate the functional condition of OFF without removing power.</p> <p>0b0010 MEM_RET. Memory Retention. Logic off with RAM retained.</p> <p>0b0011 MEM_RET_EMU. Emulated Memory Retention. Logic on with RAM on. This mode is used to emulate the functional condition of MEM_RET without removing power.</p> <p>0b1000 ON. Logic on with RAM on, cluster is functional.</p> <p>0b1001 WARM_RST. Warm Reset. Warm reset application with logic and RAM on.</p> <p>0b1010 DBG_RECOV. Debug Recovery Reset. Warm reset application with logic and RAM on.</p>	0b0000

Accessibility

Component	Offset	Range
CLUSTERPPU	0x000	None

This interface is accessible as follows:

RW

B.1.2.3.2 CLUSTERPPU_PMER, Power Mode Emulation Enable Register

This register allows software to enable entry into emulated modes.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

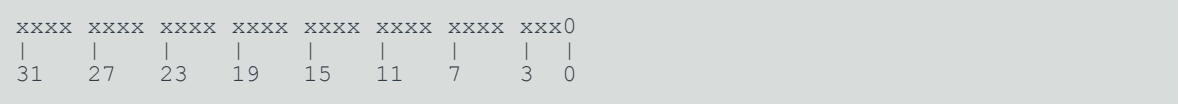
Register offset

0x004

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-70: EXT_CLUSTERPPU_PMER bit assignments

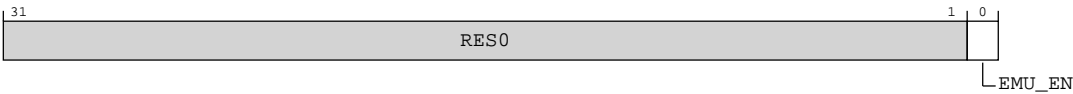


Table B-138: CLUSTERPPU_PMER bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	EMU_EN	Power mode emulation enable. 0b0 Power mode emulation disabled. 0b1 Power mode emulation enabled. Transitions to OFF and MEM_RET instead transition to OFF_EMU and MEM_RET_EMU.	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x004	None

This interface is accessible as follows:

RW

B.1.2.3.3 CLUSTERPPU_PWSR, Power Status Register

This read-only register contains status information for the power mode, operating mode, dynamic transitions, and lock feature.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x008

Access type

RO

Reset value

xxxx	xxx0	xxxx	xxxx	xxx0	xxx1	xxxx	0000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-71: EXT_CLUSTERPPU_PWSR bit assignments

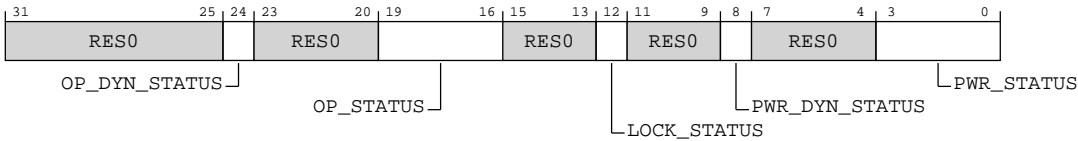


Table B-140: CLUSTERPPU_PWSR bit descriptions

Bits	Name	Description	Reset
[31:25]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[24]	OP_DYN_STATUS	<p>Operating mode dynamic transition status.</p> <p>There might be a delay in dynamic transitions becoming active or inactive if the PPU is transitioning when ext-CLUSTERPPU_PWPR.OP_DYN_EN is programmed.</p> <p>0b0</p> <p>Dynamic transitions disabled for operating modes.</p> <p>0b1</p> <p>Dynamic transitions enabled for operating modes.</p>	0b0
[23:20]	RES0	Reserved	RES0
[19:16]	OP_STATUS	<p>When L2_SLICES == 2</p> <p>Operating mode status.</p> <p>These bits reflect the current operating mode of the PPU.</p> <p>In the OFF, OFF_EMU, DBG_RECOV, and WARM_RST power modes, this field reflects the current programmed OP_POLICY even though the operating mode DEVPSTATE output bits are set to zero.</p> <p>0b0000</p> <p>OPMODE_00: The L2 Cache is not operational.</p> <p>0b0001</p> <p>OPMODE_01: Half of the L2 Cache is operational.</p> <p>0b0010</p> <p>OPMODE_02: The whole L2 Cache is operational.</p> <p>Otherwise</p> <p>Operating mode status.</p> <p>These bits reflect the current operating mode of the PPU.</p> <p>In the OFF, OFF_EMU, DBG_RECOV, and WARM_RST power modes, this field reflects the current programmed OP_POLICY even though the operating mode DEVPSTATE output bits are set to zero.</p> <p>0b0000</p> <p>OPMODE_00: The L2 Cache is not operational.</p> <p>0b0010</p> <p>OPMODE_02: The whole L2 Cache is operational.</p>	xxxx
[15:13]	RES0	Reserved	RES0
[12]	LOCK_STATUS	<p>Lock status.</p> <p>0b0</p> <p>The PPU is not locked in the current mode.</p> <p>0b1</p> <p>The PPU is locked in the current mode.</p>	0b0
[11:9]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[8]	PWR_DYN_STATUS	Power mode dynamic transition status. There might be a delay in dynamic transitions becoming active or inactive if the PPU is transitioning when ext-CLUSTERPPU_PWPR.DYN_EN is programmed. 0b0 Dynamic transitions disabled for power modes. 0b1 Dynamic transitions enabled for power modes.	0b1
[7:4]	RES0	Reserved	RES0
[3:0]	PWR_STATUS	Power mode status. These bits reflect the current power mode of the PPU. 0b0000 OFF. Logic off and RAM off. 0b0001 OFF_EMU. Emulated Off. Logic on with RAM on. This mode is used to emulate the functional condition of OFF without removing power. 0b0010 MEM_RET. Memory Retention. Logic off with RAM retained. 0b0011 MEM_RET_EMU. Emulated Memory Retention. Logic on with RAM on. This mode is used to emulate the functional condition of MEM_RET without removing power. 0b1000 ON. Logic on with RAM on, cluster is functional. 0b1001 WARM_RST. Warm Reset. Warm reset application with logic and RAM on. 0b1010 DBG_RECOV. Debug Recovery Reset. Warm reset application with logic and RAM on.	0b0000

Accessibility

Component	Offset	Range
CLUSTERPPU	0x008	None

This interface is accessible as follows:

RO

B.1.2.3.4 CLUSTERPPU_DISR, Device Interface Input Current Status Register

This read-only register contains status reflecting the values of the device interface inputs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x010

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-72: EXT_CLUSTERPPU_DISR bit assignments

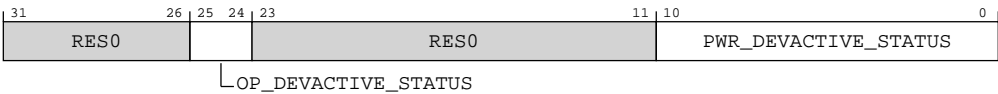


Table B-142: CLUSTERPPU_DISR bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25:24]	OP_DEVACTIVE_STATUS	Status of the operating mode DEVPACTIVE inputs. 0b00 Request for OPMODE_00. 0b01 Request for OPMODE_01. 1x Request for OPMODE_02.	0b00
[23:11]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[10:0]	PWR_DEVACTIVE_STATUS	<p>Status of the power mode DEVPACTIVE inputs.</p> <p>0b000000000000 Request for OFF.</p> <p>0000000001x Request for OFF_EMU.</p> <p>000000001xx Request for MEM_RET.</p> <p>00000001xxx Request for MEM_RET_EMU.</p> <p>001xxxxxxxx Request for ON.</p> <p>01xxxxxxxx Request for WARM_RST.</p> <p>1xxxxxxxx Request for DBG_RECOV.</p>	0b000000000000

Accessibility

Component	Offset	Range
CLUSTERPPU	0x010	None

This interface is accessible as follows:

RO

B.1.2.3.5 CLUSTERPPU_MISR, Miscellaneous Input Current Status Register

This read-only register contains status reflecting the values of miscellaneous inputs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

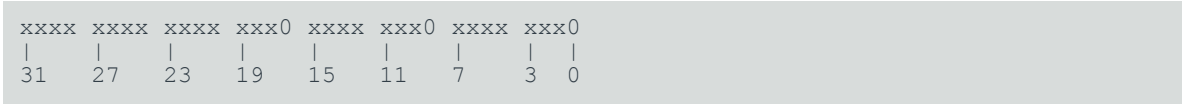
Register offset

0x014

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-73: EXT_CLUSTERPPU_MISR bit assignments

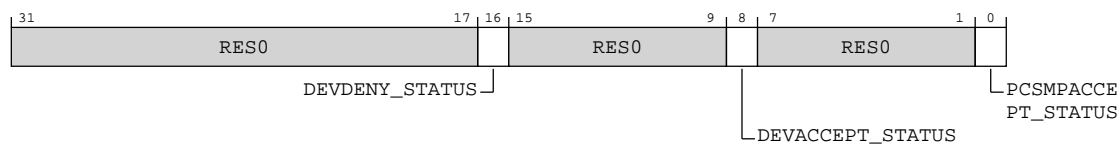


Table B-144: CLUSTERPPU_MISR bit descriptions

Bits	Name	Description	Reset
[31:17]	RES0	Reserved	RES0
[16]	DEVDPENY_STATUS	Status of the device interface DEVDPENY inputs. 0b0 DEVDPENY deasserted. 0b1 DEVDPENY asserted.	0b0
[15:9]	RES0	Reserved	RES0
[8]	DEVACCEPT_STATUS	Status of the device interface DEVPACCEPT inputs. 0b0 DEVPACCEPT deasserted. 0b1 DEVPACCEPT asserted.	0b0
[7:1]	RES0	Reserved	RES0
[0]	PCSMACCEPT_STATUS	Status of the PCSMPACCEPT inputs. 0b0 PCSMACCEPT deasserted. 0b1 PCSMACCEPT asserted.	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x014	None

This interface is accessible as follows:

RO

B.1.2.3.6 CLUSTERPPU_STSR, Stored Status Register

This register is reserved for P-channel PPU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x018

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-74: EXT_CLUSTERPPU_STSR bit assignments



Table B-146: CLUSTERPPU_STSR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x018	None

This interface is accessible as follows:

RO

B.1.2.3.7 CLUSTERPPU_UNLK, Unlock Register

This register allows software to unlock the PPU from a locked power mode.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

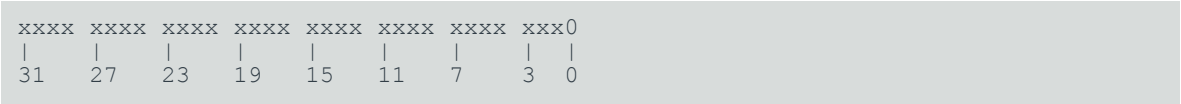
Register offset

0x01C

Access type

RAZW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-75: EXT_CLUSTERPPU_UNLK bit assignments

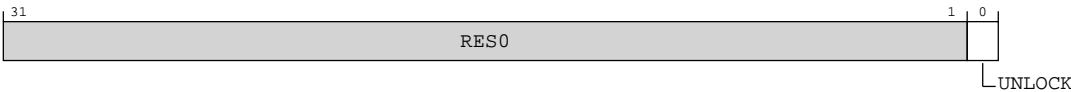


Table B-148: CLUSTERPPU_UNLK bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	UNLOCK	When 0b1 is written to this bit the PPU is unlocked from a locked power mode. A read always returns 0b0.	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x01C	None

This interface is accessible as follows:

RW

B.1.2.3.8 CLUSTERPPU_PWCR, Power Configuration Register

This register controls enabling and disabling of hardware control inputs to the PPU.



Before software programs the DEVREQEN bits it must configure the PPU for static transitions and ensure the requested power mode has been reached, this means that no further transitions can occur, otherwise behavior is UNPREDICTABLE.

The PWR_DEVACTIVEEN and OP_DEVACTIVEEN fields in this register control the ability of the DEVACTIVE inputs to initiate power mode transitions, but not the ability to generate input edge interrupt events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x020

Access type

RW

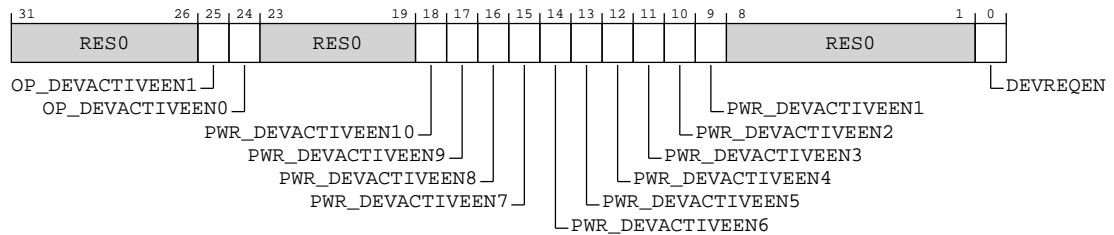
Reset value

xxxx	xx11	xxxx	x111	1111	111x	xxxx	xxx1
31	27	23	19	15	11	7	3 0

**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-76: EXT_CLUSTERPPU_PWCR bit assignments**Table B-150: CLUSTERPPU_PWCR bit descriptions**

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25]	OP_DEVACTIVEEN1	Enables the operating mode DEVPACTIVE[17] input. 0b0 DEVPACTIVE[17] input (OPMODE_02) disabled. 0b1 DEVPACTIVE[17] input (OPMODE_02) enabled.	0b1
[24]	OP_DEVACTIVEEN0	Enables the operating mode DEVPACTIVE[16] input. 0b0 DEVPACTIVE[16] input (OPMODE_01) disabled. 0b1 DEVPACTIVE[16] input (OPMODE_01) enabled.	0b1
[23:19]	RES0	Reserved	RES0
[18]	PWR_DEVACTIVEEN10	Enables the operating mode DEVPACTIVE[10] input. 0b0 DEVPACTIVE[10] input (DBG_RECOV) disabled. 0b1 DEVPACTIVE[10] input (DBG_RECOV) enabled.	0b1
[17]	PWR_DEVACTIVEEN9	Enables the operating mode DEVPACTIVE[9] input. 0b0 DEVPACTIVE[9] input (WARM_RST) disabled. 0b1 DEVPACTIVE[9] input (WARM_RST) enabled.	0b1

Bits	Name	Description	Reset
[16]	PWR_DEVACTIVEEN8	Enables the operating mode DEVACTIVE[8] input. 0b0 DEVACTIVE[8] input (ON) disabled. 0b1 DEVACTIVE[8] input (ON) enabled.	0b1
[15]	PWR_DEVACTIVEEN7	Enables the operating mode DEVACTIVE[7] input. 0b0 DEVACTIVE[7] input (unused) disabled. 0b1 DEVACTIVE[7] input (unused) enabled.	0b1
[14]	PWR_DEVACTIVEEN6	Enables the operating mode DEVACTIVE[6] input. 0b0 DEVACTIVE[6] input (unused) disabled. 0b1 DEVACTIVE[6] input (unused) enabled.	0b1
[13]	PWR_DEVACTIVEEN5	Enables the operating mode DEVACTIVE[5] input. 0b0 DEVACTIVE[5] input (unused) disabled. 0b1 DEVACTIVE[5] input (unused) enabled.	0b1
[12]	PWR_DEVACTIVEEN4	Enables the operating mode DEVACTIVE[4] input. 0b0 DEVACTIVE[4] input (unused) disabled. 0b1 DEVACTIVE[4] input (unused) enabled.	0b1
[11]	PWR_DEVACTIVEEN3	Enables the operating mode DEVACTIVE[3] input. 0b0 DEVACTIVE[3] input (MEM_RET_EMU) disabled. 0b1 DEVACTIVE[3] input (MEM_RET_EMU) enabled.	0b1
[10]	PWR_DEVACTIVEEN2	Enables the operating mode DEVACTIVE[2] input. 0b0 DEVACTIVE[2] input (MEM_RET) disabled. 0b1 DEVACTIVE[2] input (MEM_RET) enabled.	0b1
[9]	PWR_DEVACTIVEEN1	Enables the operating mode DEVACTIVE[1] input. 0b0 DEVACTIVE[1] input (OFF_EMU) disabled. 0b1 DEVACTIVE[1] input (OFF_EMU) enabled.	0b1
[8:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	DEVREQEN	Device interface handshake enable. 0b0 Device interface handshake disabled for transitions. 0b1 Device interface handshake enabled for transitions.	0b1

Accessibility

Component	Offset	Range
CLUSTERPPU	0x020	None

This interface is accessible as follows:

RW

B.1.2.3.9 CLUSTERPPU_PTCR, Power Mode Transition Register

This register contains settings which affect the behaviour of certain power mode transitions.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x024

Access type

RW

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx01
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-77: EXT_CLUSTERPPU_PTCR bit assignments

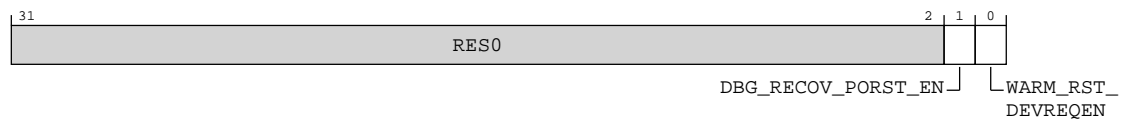


Table B-152: CLUSTERPPU_PTCR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	DBG_RECOV_PORST_EN	<p>Power-on reset behavior in DBG_RECOV.</p> <p>This bit should not be modified when the PPU is in DBG_RECOV. If it is modified, it will not have any effect on the reset output.</p> <p>This bit should not be modified when the PPU is in transition. If it is modified, the reset output will depend on either the old or the new value of the bit.</p> <p>0b0</p> <p>DEVPORESETn is not asserted when in DBG_RECOV.</p> <p>0b1</p> <p>DEVPORESETn is asserted when in DBG_RECOV.</p>	0b0
[0]	WARM_RST_DEVREQEN	<p>Device interface handshake behavior.</p> <p>This bit should not be modified when the PPU is in WARM_RST, or if the PPU is performing a transition, otherwise PPU behavior is UNPREDICTABLE.</p> <p>0b0</p> <p>The PPU does not perform a device interface handshake when transitioning between ON and WARM_RST.</p> <p>0b1</p> <p>The PPU performs a device interface handshake when transitioning between ON and WARM_RST.</p>	0b1

Accessibility

Component	Offset	Range
CLUSTERPPU	0x024	None

This interface is accessible as follows:

RW

B.1.2.3.10 CLUSTERPPU_IMR, Interrupt Mask Register

This register controls the events that assert the interrupt output. Additional event masking controls are in the Additional Interrupt Mask Register (ext-CLUSTERPPU_AIMR), Input Edge Sensitivity

Register (ext-CLUSTERPPU_IESR), and the Operating Mode Active Edge Sensitivity Register (ext-CLUSTERPPU_OPSR).

When an interrupt event is masked an occurrence of the event does not set the corresponding bit in the interrupt status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x030

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-78: EXT_CLUSTERPPU_IMR bit assignments

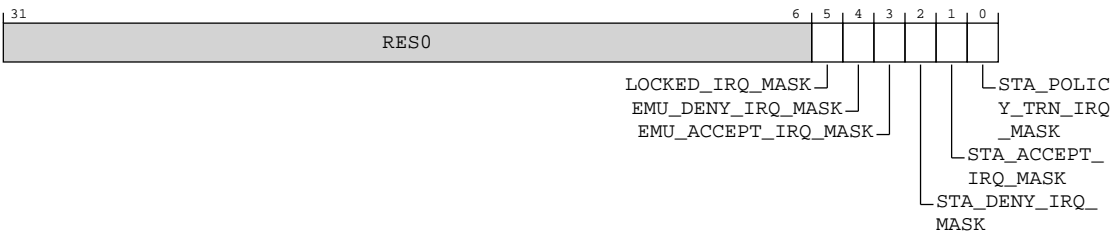


Table B-154: CLUSTERPPU_IMR bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[5]	LOCKED_IRQ_MASK	Locked event mask 0b0 Locked event enabled. 0b1 Locked event masked.	0b1
[4]	EMU_DENY_IRQ_MASK	Emulation transition denial event mask 0b0 Emulation transition denial event enabled. 0b1 Emulation transition denial event masked.	0b1
[3]	EMU_ACCEPT_IRQ_MASK	Emulation transition acceptance event mask 0b0 Emulation transition acceptance event enabled. 0b1 Emulation transition acceptance event masked.	0b1
[2]	STA_DENY_IRQ_MASK	Static transition denial event mask 0b0 Static transition denial event enabled. 0b1 Static transition denial event masked.	0b0
[1]	STA_ACCEPT_IRQ_MASK	Static transition acceptance event mask 0b0 Static transition acceptance event enabled. 0b1 Static transition acceptance event masked.	0b1
[0]	STA_POLICY_TRN_IRQ_MASK	Static full policy transition completion event mask 0b0 Static full policy transition completion event enabled. 0b1 Static full policy transition completion event masked.	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x030	None

This interface is accessible as follows:

RW

Table B-156: CLUSTERPPU_AIMR bit descriptions

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	RES0
[4]	STA_POLICY_OP_IRQ_MASK	Static operating policy transition completion event mask 0b0 Static operating policy transition completion event enabled. 0b1 Static operating policy transition completion event masked.	0b1
[3]	STA_POLICY_PWR_IRQ_MASK	Static power policy transition completion event mask 0b0 Static power policy transition completion event enabled. 0b1 Static power policy transition completion event masked.	0b1
[2]	DYN_DENY_IRQ_MASK	Dynamic transition denial event mask 0b0 Dynamic transition denial event enabled. 0b1 Dynamic transition denial event masked.	0b1
[1]	DYN_ACCEPT_IRQ_MASK	Dynamic transition acceptance event mask 0b0 Dynamic transition acceptance event enabled. 0b1 Dynamic transition acceptance event masked.	0b1
[0]	UNSPT_POLICY_IRQ_MASK	Unsupported policy event mask 0b0 Unsupported policy event enabled. 0b1 Unsupported policy event masked.	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x034	None

This interface is accessible as follows:

RW

B.1.2.3.12 CLUSTERPPU_ISR, Interrupt Status Register

This register contains information about events causing the assertion of the interrupt output. It is also used to clear interrupt events.

A bit set to 0b1 indicates the event asserted the interrupt output. Multiple events can be active at the same time. When an interrupt event is masked an occurrence of that event does not set the status bit.

A write of 0b1 to an event bit clears that event. A write of 0b0 to a bit has no effect. The interrupt output stays HIGH until all status bits in the Interrupt Status Register (CLUSTERPPU_ISR) and the Additional Interrupt Status Register (ext-CLUSTERPPU_AISR) are 0b0.

When the OTHER_IRQ bit is set, this indicates an event from the Additional Interrupt Status Register (CLUSTERPPU_AISR) has caused the interrupt output to be asserted. This bit cannot be cleared by writing to this register. It must be cleared by writing to the active event in the Additional Interrupt Status Register (ext-CLUSTERPPU_AISR).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x038

Access type

RW1C

Reset value

xxxx	xx00	xxxx	x000	xxxx	000x	0x00	0000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bits	Name	Description	Reset
[15:12]	RES0	Reserved	RES0
[11]	PWR_ACTIVE_EDGE_IRQ3	Indicates if power mode DEVPACTIVE[3] input caused the input edge event. 0b0 DEVPACTIVE[3] input (MEM_RET_EMU) did not assert the interrupt output. 0b1 DEVPACTIVE[3] input (MEM_RET_EMU) asserted the interrupt output.	0b0
[10]	PWR_ACTIVE_EDGE_IRQ2	Indicates if power mode DEVPACTIVE[2] input caused the input edge event. 0b0 DEVPACTIVE[2] input (MEM_RET) did not assert the interrupt output. 0b1 DEVPACTIVE[2] input (MEM_RET) asserted the interrupt output.	0b0
[9]	PWR_ACTIVE_EDGE_IRQ1	Indicates if power mode DEVPACTIVE[1] input caused the input edge event. 0b0 DEVPACTIVE[1] input (OFF_EMU) did not assert the interrupt output. 0b1 DEVPACTIVE[1] input (OFF_EMU) asserted the interrupt output.	0b0
[8]	RES0	Reserved	RES0
[7]	OTHER_IRQ	Indicates there is an interrupt event pending in the Additional Interrupt Status Register (ext-CLUSTERPPU_AISR). 0b0 No interrupt pending in ext-CLUSTERPPU_AISR. 0b1 Interrupt pending in ext-CLUSTERPPU_AISR.	0b0
[6]	RES0	Reserved	RES0
[5]	LOCKED_IRQ	Locked event status. 0b0 No locked event. 0b1 A locked event asserted the interrupt output.	0b0
[4]	EMU_DENY_IRQ	Emulated transition denial event status. 0b0 No emulated transition denial event. 0b1 An emulated transition denial event asserted the interrupt output.	0b0
[3]	EMU_ACCEPT_IRQ	Emulated transition acceptance event status. 0b0 No emulated transition acceptance event. 0b1 An emulated transition acceptance event asserted the interrupt output.	0b0

Bits	Name	Description	Reset
[2]	STA_DENY_IRQ	Static transition denial event status. 0b0 No static transition denial event. 0b1 An static transition denial event asserted the interrupt output.	0b0
[1]	STA_ACCEPT_IRQ	Static transition acceptance event status. 0b0 No static transition acceptance event. 0b1 An static transition acceptance event asserted the interrupt output.	0b0
[0]	STA_POLICY_TRN_IRQ	Static full policy transition completion event status. 0b0 No static full policy transition completion event. 0b1 An static full policy transition completion event asserted the interrupt output.	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x038	None

This interface is accessible as follows:

RW

B.1.2.3.13 CLUSTERPPU_AISR, Additional Interrupt Status Register

This register contains information about events causing the assertion of the interrupt output. It is also used to clear interrupt events.

A bit set to 0b1 indicates the event asserted the interrupt output. Multiple events can be active at the same time. When an interrupt event is masked an occurrence of that event does not set the status bit.

A write of 0b1 to an event bit clears that event. A write of 0b0 has no effect. The interrupt output stays HIGH until all status bits in the Interrupt Status Register (ext-CLUSTERPPU_ISR) and the Additional Interrupt Status Register (CLUSTERPPU_AISR) are set to 0b0.

When an interrupt status is set to 0b1 in this register it sets the OTHER_IRQ bit in the Interrupt Status Register (ext-CLUSTERPPU_ISR). Status bits in this register are only cleared by writing to this register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x03C

Access type

RW1C

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-81: EXT_CLUSTERPPU_AISR bit assignments

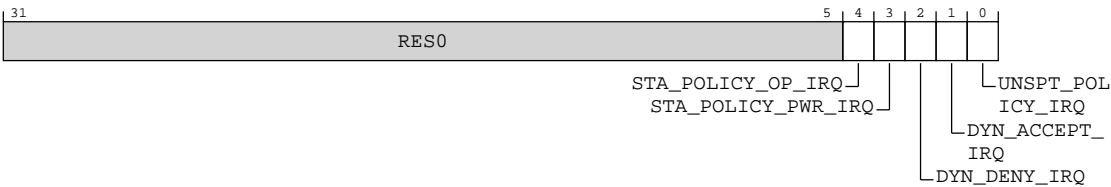


Table B-160: CLUSTERPPU_AISR bit descriptions

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	RES0
[4]	STA_POLICY_OP_IRQ	Static operating policy transition completion event status 0b0 No static operating policy transition completion event. 0b1 A static operating policy transition completion event asserted the interrupt output.	0b0
[3]	STA_POLICY_PWR_IRQ	Static power policy transition completion event status 0b0 No static power policy transition completion event. 0b1 A static power policy transition completion event asserted the interrupt output.	0b0

Bits	Name	Description	Reset
[2]	DYN_DENY_IRQ	Dynamic transition denial event status 0b0 No dynamic transition denial event. 0b1 A dynamic transition denial event asserted the interrupt output.	0b0
[1]	DYN_ACCEPT_IRQ	Dynamic transition acceptance event status 0b0 No dynamic transition acceptance event. 0b1 A dynamic transition acceptance event asserted the interrupt output.	0b0
[0]	UNSPT_POLICY_IRQ	Unsupported policy event status 0b0 No unsupported policy event. 0b1 An unsupported policy event asserted the interrupt output.	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x03C	None

This interface is accessible as follows:

RW

B.1.2.3.14 CLUSTERPPU_IESR, Input Edge Sensitivity Register

This register configures the transitions on the power mode DEVPACTIVE inputs that generate an Input Edge interrupt event.

When an event is masked an occurrence of the event does not set the corresponding bit in the interrupt status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x040

Access type

RW

Reset value

xxxx	xxxx	xx00	0000	0000	0000	0000	00xx
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-82: EXT_CLUSTERPPU_IESR bit assignments

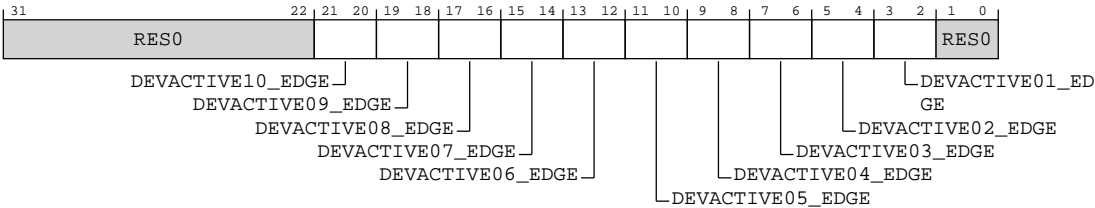


Table B-162: CLUSTERPPU_IESR bit descriptions

Bits	Name	Description	Reset
[31:22]	RES0	Reserved	RES0
[21:20]	DEVACTIVE10_EDGE	Configures the transitions on the DEVPACTIVE[10] input (DBG_RECOV) that generate an Input Edge interrupt event. 0b00 Event maksed. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00

Bits	Name	Description	Reset
[19:18]	DEVACTIVE09_EDGE	<p>Configures the transitions on the DEVPACTIVE[9] input (WARM_RST) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00
[17:16]	DEVACTIVE08_EDGE	<p>Configures the transitions on the DEVPACTIVE[8] input (ON) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00
[15:14]	DEVACTIVE07_EDGE	<p>Configures the transitions on the DEVPACTIVE[7] input (unused) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00
[13:12]	DEVACTIVE06_EDGE	<p>Configures the transitions on the DEVPACTIVE[6] input (unused) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00

Bits	Name	Description	Reset
[11:10]	DEVACTIVE05_EDGE	<p>Configures the transitions on the DEVPACTIVE[5] input (unused) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00
[9:8]	DEVACTIVE04_EDGE	<p>Configures the transitions on the DEVPACTIVE[4] input (unused) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00
[7:6]	DEVACTIVE03_EDGE	<p>Configures the transitions on the DEVPACTIVE[3] input (MEM_RET_EMU) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00
[5:4]	DEVACTIVE02_EDGE	<p>Configures the transitions on the DEVPACTIVE[2] input (MEM_RET) that generate an Input Edge interrupt event.</p> <p>0b00 Event masked.</p> <p>0b01 Rising edge of event generates an interrupt.</p> <p>0b10 Falling edge of event generates an interrupt.</p> <p>0b11 Both edges of event generate an interrupt.</p>	0b00

Bits	Name	Description	Reset
[3:2]	DEVACTIVE01_EDGE	Configures the transitions on the DEVPACTIVE[1] input (OFF_EMU) that generate an Input Edge interrupt event. 0b00 Event masked. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00
[1:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x040	None

This interface is accessible as follows:

RW

B.1.2.3.15 CLUSTERPPU_OPSR, Operating Mode Active Edge Sensitivity Register

This register configures the transitions on the operating mode DEVPACTIVE inputs that generate an Input Edge interrupt event.

When an event is masked an occurrence of the event does not set the corresponding bit in the interrupt status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x044

Access type

RW

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-83: EXT_CLUSTERPPU_OPSR bit assignments

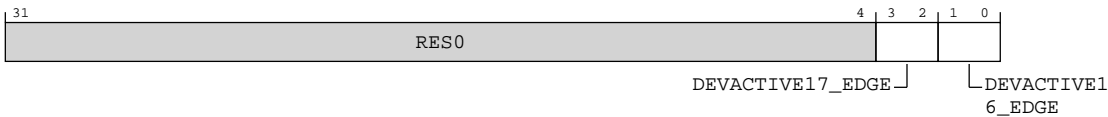


Table B-164: CLUSTERPPU_OPSR bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:2]	DEVACTIVE17_EDGE	Configures the transitions on the DEVPACTIVE[17] input (OPMODE_02) that generate an Input Edge interrupt event. 0b00 Event maksed. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00
[1:0]	DEVACTIVE16_EDGE	Configures the transitions on the DEVPACTIVE[16] input (OPMODE_01) that generate an Input Edge interrupt event. 0b00 Event maksed. 0b01 Rising edge of event generates an interrupt. 0b10 Falling edge of event generates an interrupt. 0b11 Both edges of event generate an interrupt.	0b00

Accessibility

Component	Offset	Range
CLUSTERPPU	0x044	None

This interface is accessible as follows:

RW

B.1.2.3.16 CLUSTERPPU_FUNRR, Functional Retention RAM Configuration Register

This register is reserved.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

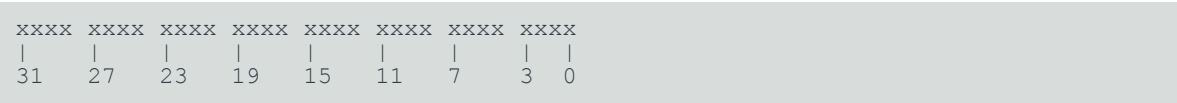
Register offset

0x050

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-84: EXT_CLUSTERPPU_FUNRR bit assignments

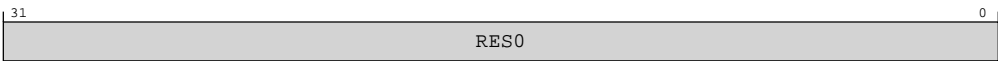


Table B-166: CLUSTERPPU_FUNRR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x050	None

This interface is accessible as follows:

RW

B.1.2.3.17 CLUSTERPPU_FULRR, Full Retention RAM Configuration Register

This register is reserved (high bits of PCSMPSTATE outputs are not configurable by software).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

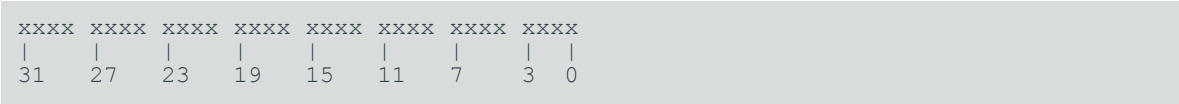
Register offset

0x054

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-85: EXT_CLUSTERPPU_FULRR bit assignments

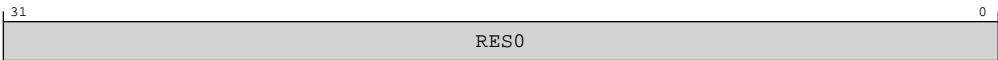


Table B-168: CLUSTERPPU_FULRR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x054	None

This interface is accessible as follows:

RW

B.1.2.3.18 CLUSTERPPU_MEMRR, Memory Retention RAM Configuration Register

This register is reserved (high bits of PCSMPSTATE outputs are not configurable by software).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0x058

Access type

RW

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-86: EXT_CLUSTERPPU_MEMRR bit assignments

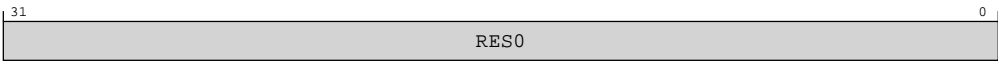


Table B-170: CLUSTERPPU_MEMRR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x058	None

This interface is accessible as follows:

RW

B.1.2.3.19 CLUSTERPPU_EDTR0, Power Mode Entry Delay Register 0

This register is reserved (power mode entry delays not supported).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

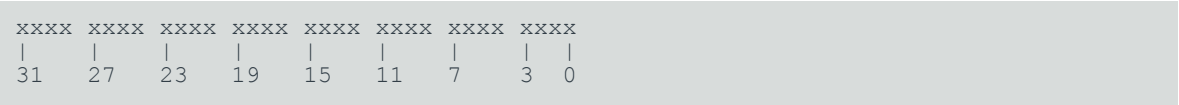
Register offset

0x160

Access type

RW

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-87: EXT_CLUSTERPPU_EDTR0 bit assignments

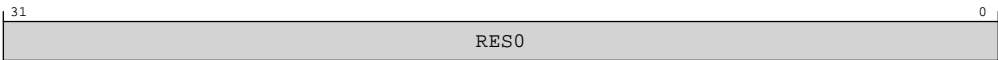


Table B-172: CLUSTERPPU_EDTR0 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x160	None

This interface is accessible as follows:

RW

B.1.2.3.20 CLUSTERPPU_EDTR1, Power Mode Entry Delay Register 1

This register is reserved (power mode entry delays not supported).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

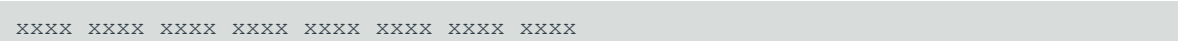
Register offset

0x164

Access type

RW

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-88: EXT_CLUSTERPPU_EDTR1 bit assignments

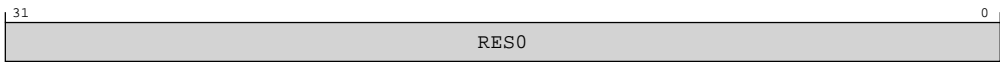


Table B-174: CLUSTERPPU_EDTR1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0x164	None

This interface is accessible as follows:

RW

B.1.2.3.21 CLUSTERPPU_DCDR0, Device Control Delay Configuration Register 0

This register is used to program device control delay parameters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

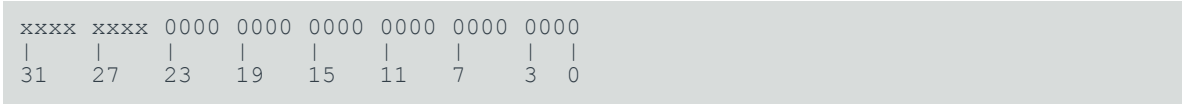
Register offset

0x170

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-89: EXT_CLUSTERPPU_DCDR0 bit assignments

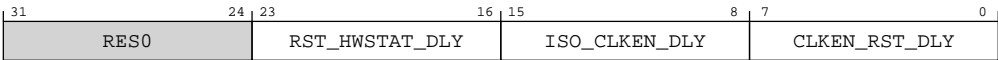


Table B-176: CLUSTERPPU_DCDR0 bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:16]	RST_HWSTAT_DLY	Delay from reset de-assertion to HWSTAT update.	0x00
[15:8]	ISO_CLKEN_DLY	Delay from isolation enable de-assertion to clock enable assertion.	0x00
[7:0]	CLKEN_RST_DLY	Delay from clock enable assertion to reset de-assertion.	0x00

Accessibility

Component	Offset	Range
CLUSTERPPU	0x170	None

This interface is accessible as follows:

RW

B.1.2.3.22 CLUSTERPPU_DCDR1, Device Control Delay Configuration Register 1

This register is used to program device control delay parameters.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
CLUSTERPPU

Register offset
0x174

Access type
RW

Reset value

xxxx	xxxx	xxxx	xxxx	0000	0000	0000	0000
31	27	23	19	15	11	7	3
							0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-90: EXT_CLUSTERPPU_DCDR1 bit assignments

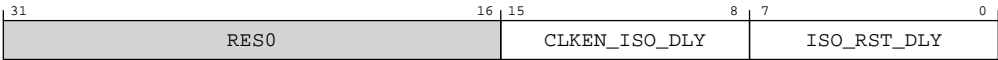


Table B-178: CLUSTERPPU_DCDR1 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:8]	CLKEN_ISO_DLY	Delay from clock enable de-assertion to isolation enable assertion.	0x00
[7:0]	ISO_RST_DLY	Delay from isolation enable assertion to reset assertion.	0x00

Accessibility

Component	Offset	Range
CLUSTERPPU	0x174	None

This interface is accessible as follows:

RW

Table B-180: CLUSTERPPU_IDR0 bit descriptions

Bits	Name	Description	Reset
[31:30]	RES0	Reserved	RES0
[29]	DYN_WRM_RST_SPT	Dynamic WARM_RST support. 0b0 Dynamic WARM_RST not supported.	0b0
[28]	DYN_ON_SPT	Dynamic ON support. 0b1 Dynamic ON supported.	0b1
[27]	DYN_FUNC_RET_SPT	Dynamic DYN_FUNC_RET_SPT support. 0b0 Dynamic DYN_FUNC_RET_SPT not supported.	0b0
[26]	DYN_FULL_RET_SPT	Dynamic DYN_FULL_RET_SPT support. 0b0 Dynamic DYN_FULL_RET_SPT not supported.	0b0
[25]	DYN_MEM_OFF_SPT	Dynamic MEM_OFF support. 0b0 Dynamic MEM_OFF not supported.	0b0
[24]	DYN_LGC_RET_SPT	Dynamic LOGIC_RET support. 0b0 Dynamic LOGIC_RET not supported.	0b0
[23]	DYN_MEM_RET_EMU_SPT	Dynamic DYN_MEM_RET_EMU_SPT support. 0b1 Dynamic DYN_MEM_RET_EMU_SPT supported.	0b1
[22]	DYN_MEM_RET_SPT	Dynamic DYN_MEM_RET_SPT support. 0b1 Dynamic DYN_MEM_RET_SPT supported.	0b1
[21]	DYN_OFF_EMU_SPT	Dynamic OFF_EMU support. 0b1 Dynamic OFF_EMU supported.	0b1
[20]	DYN_OFF_SPT	Dynamic OFF support. 0b1 Dynamic OFF supported.	0b1
[19]	RES0	Reserved	RES0
[18]	STA_DBG_RECOV_SPT	DBG_RECOV support. 0b1 DBG_RECOV supported.	0b1
[17]	STA_WRM_RST_SPT	WRM_RST support. 0b1 WRM_RST supported.	0b1

Bits	Name	Description	Reset
[16]	STA_ON_SPT	ON support. 0b1 ON supported.	0b1
[15]	STA_FUNC_RET_SPT	FUNC_RET support. 0b0 FUNC_RET not supported.	0b0
[14]	STA_FULL_RET_SPT	FULL_RET support. 0b0 FULL_RET not supported.	0b0
[13]	STA_MEM_OFF_SPT	MEM_OFF support. 0b0 MEM_OFF not supported.	0b0
[12]	STA_LGC_RET_SPT	LOGIC_RET support. 0b0 LOGIC_RET not supported.	0b0
[11]	STA_MEM_RET_EMU_SPT	MEM_RET_EMU support. 0b1 MEM_RET_EMU supported.	0b1
[10]	STA_MEM_RET_SPT	MEM_RET support. 0b1 MEM_RET supported.	0b1
[9]	STA_OFF_EMU_SPT	OFF_EMU support. 0b1 OFF_EMU supported.	0b1
[8]	STA_OFF_SPT	OFF support. 0b1 OFF supported.	0b1
[7:4]	NUM_OPMODE	No. of operating modes supported, minus 1. 0b0010 3 operating modes supported.	0b0010
[3:0]	DEVCHAN	No. of Device Interface Channels. 0b0000 0 (P-channel PPU).	0b0000

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFB0	None

This interface is accessible as follows:

RO

B.1.2.3.24 CLUSTERPPU_IDR1, PPU Identification Register 1

This read-only register contains information on the optional features and configurations that are supported by this CLUSTERPPU.

Additional information on optional features can be found in the PPU Identification Register 0 (ext-CLUSTERPPU_IDR0).

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFB4

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxx1	x011	x000	x110
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-92: EXT_CLUSTERPPU_IDR1 bit assignments

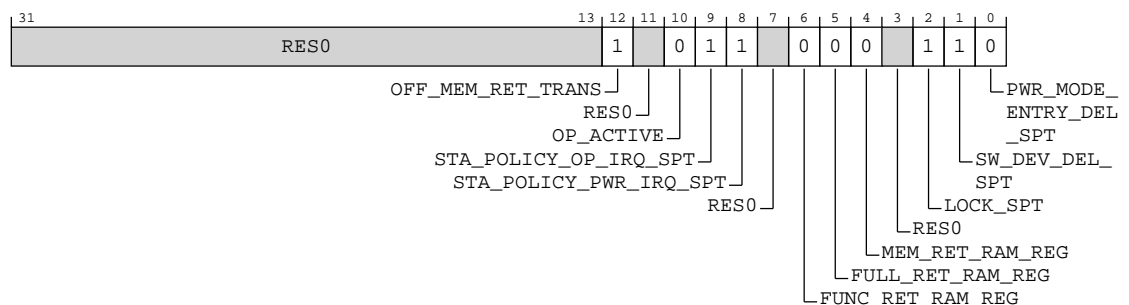


Table B-182: CLUSTERPPU_IDR1 bit descriptions

Bits	Name	Description	Reset
[31:13]	RES0	Reserved	RES0
[12]	OFF_MEM_RET_TRANS	<p>OFF to MEM_RET direct transition. Indicates if direct transitions from OFF to MEM_RET and from OFF_EMU to MEM_RET_EMU are supported.</p> <p>0b1</p> <p>OFF to MEM_RET direct transition supported.</p>	0b1
[11]	RES0	Reserved	RES0
[10]	OP_ACTIVE	<p>Operating mode use model for dynamic transitions.</p> <p>0b0</p> <p>Ladder use model.</p>	0b0
[9]	STA_POLICY_OP_IRQ_SPT	<p>Operating policy transition completion event status.</p> <p>0b1</p> <p>Operating policy transition completion events supported.</p>	0b1
[8]	STA_POLICY_PWR_IRQ_SPT	<p>Power policy transition completion event status.</p> <p>0b1</p> <p>Power policy transition completion events supported.</p>	0b1
[7]	RES0	Reserved	RES0
[6]	FUNC_RET_RAM_REG	<p>Indicates if the ext-CLUSTERPPU_FUNRR register is present or reserved.</p> <p>0b0</p> <p>ext-CLUSTERPPU_FUNRR is reserved.</p>	0b0
[5]	FULL_RET_RAM_REG	<p>Indicates if the ext-CLUSTERPPU_FULRR register is present or reserved.</p> <p>0b0</p> <p>ext-CLUSTERPPU_FULRR is reserved.</p>	0b0
[4]	MEM_RET_RAM_REG	<p>Indicates if the ext-CLUSTERPPU_MEMRR register is present or reserved.</p> <p>0b0</p> <p>ext-CLUSTERPPU_MEMRR is not present.</p>	0b0
[3]	RES0	Reserved	RES0
[2]	LOCK_SPT	<p>Indicates if the lock and the lock interrupt event are supported.</p> <p>0b1</p> <p>Lock and the lock interrupt event are supported.</p>	0b1
[1]	SW_DEV_DEL_SPT	<p>Software device delay control configuration support.</p> <p>0b1</p> <p>Software device delay control configuration supported.</p>	0b1
[0]	PWR_MODE_ENTRY_DEL_SPT	<p>Power mode entry delay support.</p> <p>0b0</p> <p>Power mode entry delay not supported.</p>	0b0

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFB4	None

This interface is accessible as follows:

RO

B.1.2.3.25 CLUSTERPPU_IIDR, Implementation Identification Register

This register provides information about the implementer and implementation of the PPU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFC8

Access type

RO

Reset value

0000 1011 0110 0001 0001 0100 0011 1011

Bit descriptions

Figure B-93: EXT_CLUSTERPPU_IIDR bit assignments

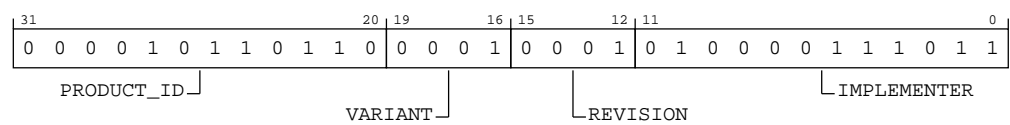


Table B-184: CLUSTERPPU_IIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	PRODUCT_ID	Value identifying the PPU part. 0b000010110110 PCK-600 PPU.	0x0B6
[19:16]	VARIANT	Value used to distinguish product variants, or major revisions of the product. 0b0001 Product variant r0p5.	0b0001
[15:12]	REVISION	Value used to distinguish minor revisions of the product. 0b0001 No ECO fixes.	0b0001

Bits	Name	Description	Reset
[11:0]	IMPLEMENTER	Implementer identification. 0b010000111011 Arm Limited.	0x43B

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFC8	None

This interface is accessible as follows:

RO

B.1.2.3.26 CLUSTERPPU_AIDR, Architecture Identification Register

This register identifies the PPU architecture revision.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFCC

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	0001
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-94: EXT_CLUSTERPPU_AIDR bit assignments

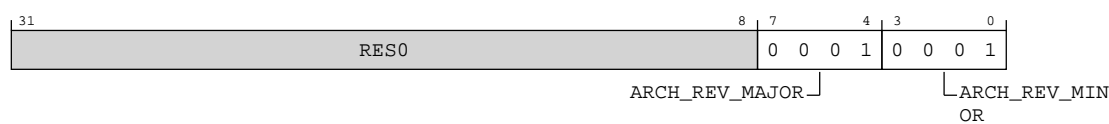


Table B-186: CLUSTERPPU_AIDR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	ARCH_REV_MAJOR	PPU architecture major revision. 0b0001 PPU architecture major revision 1.	0b0001
[3:0]	ARCH_REV_MINOR	PPU architecture minor revision. 0b0001 PPU architecture minor revision 1.	0b0001

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFCC	None

This interface is accessible as follows:

RO

B.1.2.3.27 CLUSTERPPU_PIDR4, PPU Peripheral Identification Register 4

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

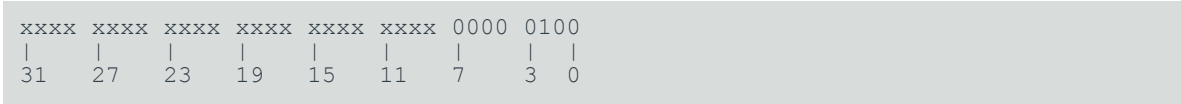
Register offset

0xFD0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-95: EXT_CLUSTERPPU_PIDR4 bit assignments



Table B-188: CLUSTERPPU_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
CLUSTERPPU	0xFD0	None

This interface is accessible as follows:

RO

B.1.2.3.28 CLUSTERPPU_PIDR5, PPU Peripheral Identification Register 5

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
CLUSTERPPU

Register offset
0xFD4

Access type
RO

Reset value

xxxx

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-96: EXT_CLUSTERPPU_PIDR5 bit assignments

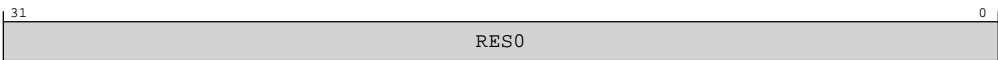


Table B-190: CLUSTERPPU_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFD4	None

This interface is accessible as follows:

RO

B.1.2.3.29 CLUSTERPPU_PIDR6, PPU Peripheral Identification Register 6

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFD8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-97: EXT_CLUSTERPPU_PIDR6 bit assignments

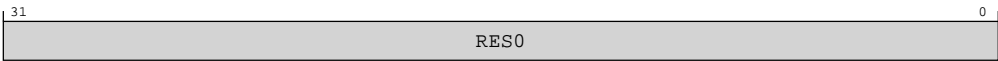


Table B-192: CLUSTERPPU_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFD8	None

This interface is accessible as follows:

RO

B.1.2.3.30 CLUSTERPPU_PIDR7, PPU Peripheral Identification Register 7

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFDC

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-98: EXT_CLUSTERPPU_PIDR7 bit assignments

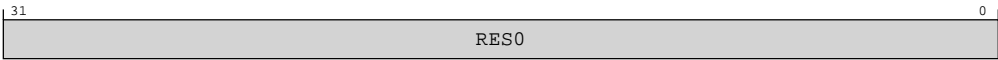


Table B-194: CLUSTERPPU_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFDC	None

This interface is accessible as follows:

RO

B.1.2.3.31 CLUSTERPPU_PIDR0, PPU Peripheral Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFE0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-99: EXT_CLUSTERPPU_PIDR0 bit assignments

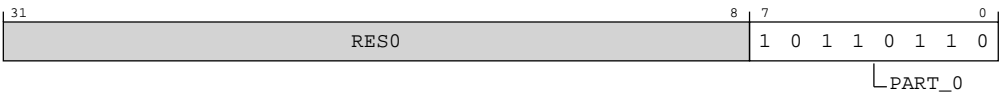


Table B-196: CLUSTERPPU_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PART_0	Part number bits [7:0]. 0b10110110 PCK-600 PPU. Bits [7:0] of part number 0x0B6.	0xB6

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFE0	None

This interface is accessible as follows:

RO

B.1.2.3.32 CLUSTERPPU_PIDR1, PPU Peripheral Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFE4

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1011	0000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-100: EXT_CLUSTERPPU_PIDR1 bit assignments

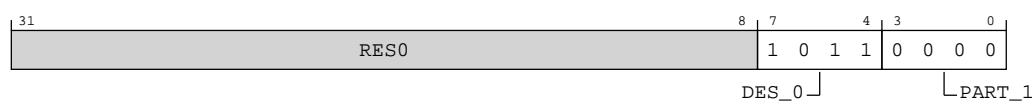


Table B-198: CLUSTERPPU_PIDR1 bit descriptions

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]. 0b0000 PCK-600 PPU. Bits [11:8] of part number 0x0B6.	0b0000

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFE4	None

This interface is accessible as follows:

RO

B.1.2.3.33 CLUSTERPPU_PIDR2, PPU Peripheral Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFE8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-101: EXT_CLUSTERPPU_PIDR2 bit assignments

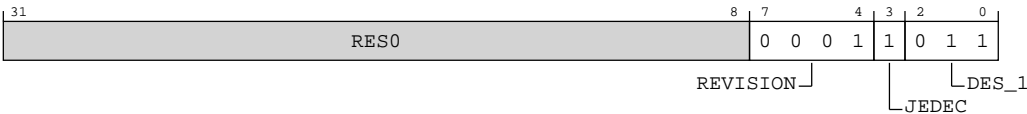


Table B-200: CLUSTERPPU_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0001 Revision r0p5.	0b0001
[3]	JEDEC	JEDEC assignee. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	JEP106 identification code bits [6:4]. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFE8	None

This interface is accessible as follows:

RO

B.1.2.3.34 CLUSTERPPU_PIDR3, PPU Peripheral Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFEC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-102: EXT_CLUSTERPPU_PIDR3 bit assignments

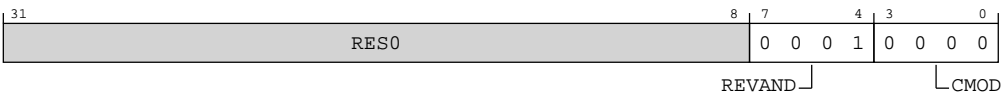


Table B-202: CLUSTERPPU_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Minor errata fixes. 0b0001 No ECO fixes.	0b0001
[3:0]	CMOD	Customer Modified. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFEC	None

This interface is accessible as follows:

RO

B.1.2.3.35 CLUSTERPPU_CIDR0, CLUSTERPPU Component Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-103: EXT_CLUSTERPPU_CIDR0 bit assignments



Table B-204: CLUSTERPPU_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
CLUSTERPPU	0xFF0	None

This interface is accessible as follows:

RO

B.1.2.3.36 CLUSTERPPU_CIDR1, CLUSTERPPU Component Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

Register offset

0xFF4

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1111	0000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-104: EXT_CLUSTERPPU_CIDR1 bit assignments

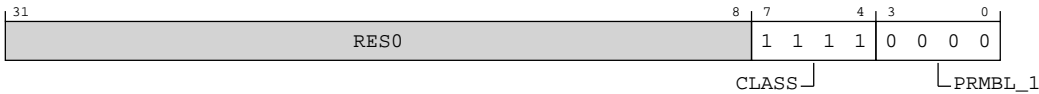


Table B-206: CLUSTERPPU_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. 0b1111 CoreLink component.	0b1111
[3:0]	PRMBL_1	CoreSight component identification preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFF4	None

This interface is accessible as follows:

RO

B.1.2.3.37 CLUSTERPPU_CIDR2, CLUSTERPPU Component Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

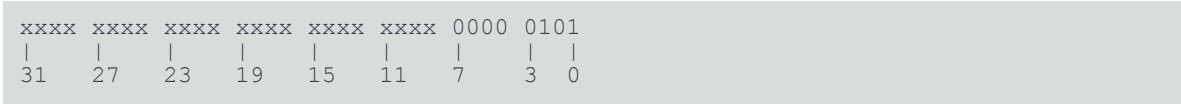
Register offset

0xFF8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-105: EXT_CLUSTERPPU_CIDR2 bit assignments



Table B-208: CLUSTERPPU_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	CoreSight component identification preamble. 0b000000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
CLUSTERPPU	0xFF8	None

This interface is accessible as follows:

RO

B.1.2.3.38 CLUSTERPPU_CIDR3, CLUSTERPPU Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPPU

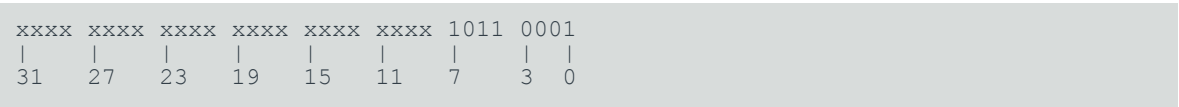
Register offset

0xFFC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-106: EXT_CLUSTERPPU_CIDR3 bit assignments



Table B-210: CLUSTERPPU_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
CLUSTERPPU	0xFFC	None

This interface is accessible as follows:

RO

B.2 Registers accessed over the Debug APB bus

This section contains the summary and descriptions for all the external registers in the Cortex®-R82 processor accessed over the Debug APB bus.

Debug APB bus implements one of two memory maps:

- A sparse memory map, when the configuration parameter `DENSE_CS_ADDR_MAP` is 0. Each set of System registers is grouped on separate 64KB page boundaries. This allows to isolate and remap components when using the EL1 MMU (if VMSA included) with the maximum 64KB granule.
- A dense memory map, when the configuration parameter `DENSE_CS_ADDR_MAP` is 1. Each set of System registers is grouped on separate 4KB page boundaries. It may not be possible to isolate and remap individual components using an EL1 MMU with 16KB or 64KB granules. You can either use a 4KB granule, or depend on an EL2 MPU (or an EL1 MPU, if EL1 is using PMSA) to isolate components without remapping.

The following table shows the total space occupied by the Debug APB bus memory map, depending on how many cores are implemented in the processor.

Table B-212: Debug APB bus sparse and dense memory map sizes

Number of cores	Sparse memory map	Dense memory map
1	24-bit (16MB)	16-bit (64KB)
2	24-bit (16MB)	17-bit (128KB)
3	24-bit (16MB)	17-bit (128KB)
4	24-bit (16MB)	17-bit (128KB)
5	24-bit (16MB)	18-bit (256KB)
6	24-bit (16MB)	18-bit (256KB)
7	24-bit (16MB)	18-bit (256KB)
8	24-bit (16MB)	18-bit (256KB)

The following table shows the base addresses for each set of system component registers that the external agents can access using the Debug APB bus.



Note

The Debug APB memory map for the Cortex®-R82 processor depends on the specific implementation of your cluster, for example the number of cores configured in the cluster or if the *Embedded Logic Analyzer* (ELA) included or not.

Table B-213: Memory map for external agents accessing the Debug APB bus

Base address, sparse map (<code>DENSE_CS_ADDR_MAP</code> = 0)	Base address, dense map (<code>DENSE_CS_ADDR_MAP</code> = 1)	Registers	Memory map
0x000000	0x00000	DebugBlock ROM Table	B.2.1.3 External DBROM registers summary on page 1604
0x100000	0x0A000	Core 0 CTI	B.2.1.6 External CTI registers summary on page 1608

Base address, sparse map (DENSE_CS_ADDR_MAP = 0)	Base address, dense map (DENSE_CS_ADDR_MAP = 1)	Registers	Memory map
0x110000	0x10000	Core 1 CTI	B.2.1.6 External CTI registers summary on page 1608
0x120000	0x16000	Core 2 CTI	B.2.1.6 External CTI registers summary on page 1608
0x130000	0x1C000	Core 3 CTI	B.2.1.6 External CTI registers summary on page 1608
0x140000	0x22000	Core 4 CTI	B.2.1.6 External CTI registers summary on page 1608
0x150000	0x28000	Core 5 CTI	B.2.1.6 External CTI registers summary on page 1608
0x160000	0x2E000	Core 6 CTI	B.2.1.6 External CTI registers summary on page 1608
0x170000	0x34000	Core 7 CTI	B.2.1.6 External CTI registers summary on page 1608
0x200000	0x01000	Cluster ROM Table	B.2.1.4 External CLUSTERROM registers summary on page 1605
0x210000	0x02000	Cluster PMU	B.2.1.2 External CLUSTERPMU registers summary on page 1602
0x220000	0x03000	Cluster ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual
0x230000	0x04000	Cluster CTI	B.2.1.6 External CTI registers summary on page 1608
0x800000	0x05000	Core 0 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0x810000	0x06000	Core 0 Debug	B.2.1.1 External Debug registers summary on page 1601
0x820000	0x07000	Core 0 PMU	B.2.1.8 External PMU registers summary on page 1613
0x830000	0x08000	Core 0 ETM	B.2.1.7 External ETM registers summary on page 1609
0x840000	0x09000	Core 0 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual
0x900000	0x0B000	Core 1 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0x910000	0x0C000	Core1 Debug	B.2.1.1 External Debug registers summary on page 1601
0x920000	0x0D000	Core 1 PMU	B.2.1.8 External PMU registers summary on page 1613
0x930000	0x0E000	Core 1 ETM	B.2.1.7 External ETM registers summary on page 1609
0x940000	0x0F000	Core 1 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual
0xA00000	0x11000	Core 2 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0xA10000	0x12000	Core 2 Debug	B.2.1.1 External Debug registers summary on page 1601

Base address, sparse map (DENSE_CS_ADDR_MAP = 0)	Base address, dense map (DENSE_CS_ADDR_MAP = 1)	Registers	Memory map
0xA20000	0x13000	Core 2 PMU	B.2.1.8 External PMU registers summary on page 1613
0xA30000	0x14000	Core 2 ETM	B.2.1.7 External ETM registers summary on page 1609
0xA40000	0x15000	Core 2 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual
0xB00000	0x17000	Core 3 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0xB10000	0x18000	Core 3 Debug	B.2.1.1 External Debug registers summary on page 1601
0xB20000	0x19000	Core 3 PMU	B.2.1.8 External PMU registers summary on page 1613
0xB30000	0x1A000	Core 3 ETM	B.2.1.7 External ETM registers summary on page 1609
0xB40000	0x1B000	Core 3 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual
0xC00000	0x1D000	Core 4 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0xC10000	0x1E000	Core 4 Debug	B.2.1.1 External Debug registers summary on page 1601
0xC20000	0x1F000	Core 4 PMU	B.2.1.8 External PMU registers summary on page 1613
0xC30000	0x20000	Core4 ETM	B.2.1.7 External ETM registers summary on page 1609
0xC40000	0x21000	Core 4 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual
0xD00000	0x23000	Core 5 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0xD10000	0x24000	Core 5 Debug	B.2.1.1 External Debug registers summary on page 1601
0xD20000	0x25000	Core5 PMU	B.2.1.8 External PMU registers summary on page 1613
0xD30000	0x26000	Core5 ETM	B.2.1.7 External ETM registers summary on page 1609
0xD40000	0x27000	Core5 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual
0xE00000	0x29000	Core 6 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0xE10000	0x2A000	Core 6 Debug	B.2.1.1 External Debug registers summary on page 1601
0xE20000	0x2B000	Core 6 PMU	B.2.1.8 External PMU registers summary on page 1613
0xE30000	0x2C000	Core 6 ETM	B.2.1.7 External ETM registers summary on page 1609
0xE40000	0x2D000	Core6 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual

Base address, sparse map (DENSE_CS_ADDR_MAP = 0)	Base address, dense map (DENSE_CS_ADDR_MAP = 1)	Registers	Memory map
0xF00000	0x2F000	Core 7 ROM Table	B.2.1.5 External COREROM registers summary on page 1607
0xF10000	0x30000	Core 7 Debug	B.2.1.1 External Debug registers summary on page 1601
0xF20000	0x31000	Core 7 PMU	B.2.1.8 External PMU registers summary on page 1613
0xF30000	0x32000	Core 7 ETM	B.2.1.7 External ETM registers summary on page 1609
0xF40000	0x33000	Core 7 ELA	See Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual

B.2.1 Register summaries for registers accessed over the Debug APB bus

This section includes the summary tables for all the external registers in the Cortex®-R82 processor accessed over the Debug APB bus.

B.2.1.1 External Debug registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-214: Debug registers summary

Offset	Name	Reset	Width	Description
0x020	EDES	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_ELO	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_ELO	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 + (16 * n)	DBGBVR<n>_EL1 [63:0]	See individual bit resets.	64-bit	Debug Breakpoint Value Registers

Offset	Name	Reset	Width	Description
0x408 + (16 * n)	DBGBCR<n>_EL1	See individual bit resets.	64-bit	Debug Breakpoint Control Registers
0x800 + (16 * n)	DBGWVR<n>_EL1 [63:0]	See individual bit resets.	64-bit	Debug Watchpoint Value Registers
0x808 + (16 * n)	DBGWCR<n>_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 [31:0]	See individual bit resets.	32-bit	External Debug Processor Feature Register
0xD24	EDPFR [63:32]	See individual bit resets.	32-bit	External Debug Processor Feature Register
0xD28	EDDFR [31:0]	See individual bit resets.	32-bit	External Debug Feature Register
0xD2C	EDDFR [63:32]	See individual bit resets.	32-bit	External Debug Feature Register
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.2.1.2 External CLUSTERPMU registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped CLUSTERPMU 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-215: CLUSTERPMU registers summary

Offset	Name	Reset	Width	Description
0x000 + (8 * n)	CLUSTERPMU_PMEVCNTR<n>_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Event Count Registers
0x0F8	CLUSTERPMU_PMCCNTR_EL1 [31:0]	See individual bit resets.	32-bit	Cluster Performance Monitors Cycle Counter
0x0FC	CLUSTERPMU_PMCCNTR_EL1 [63:32]	See individual bit resets.	32-bit	Cluster Performance Monitors Cycle Counter
0x400 + (4 * n)	CLUSTERPMU_PMEVTYPER<n>_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Event Type Registers
0x47C	CLUSTERPMU_PMCCFILTR_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Cycle Counter Filter Register
0xC00	CLUSTERPMU_PMCNTENSET_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Count Enable Set register
0xC20	CLUSTERPMU_PMCNTENCLR_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Count Enable Clear register
0xC40	CLUSTERPMU_PMINTENSET_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Interrupt Enable Set register
0xC60	CLUSTERPMU_PMINTENCLR_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Interrupt Enable Clear register
0xC80	CLUSTERPMU_PMOVSLR_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Overflow Flag Status Clear register
0xCA0	CLUSTERPMU_PMSWINC_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Software Increment register
0xCC0	CLUSTERPMU_PMOVSSSET_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Overflow Flag Status Set register
0xE00	CLUSTERPMU_PMCFGR	See individual bit resets.	32-bit	Cluster Performance Monitors Configuration Register
0xE04	CLUSTERPMU_PMCER_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Control Register
0xE20	CLUSTERPMU_PMCEID0	See individual bit resets.	32-bit	Cluster Performance Monitors Common Event Identification register 0
0xE24	CLUSTERPMU_PMCEID1	See individual bit resets.	32-bit	Cluster Performance Monitors Common Event Identification register 1
0xE28	CLUSTERPMU_PMCEID2	See individual bit resets.	32-bit	Cluster Performance Monitors Common Event Identification register 2
0xE2C	CLUSTERPMU_PMCEID3	See individual bit resets.	32-bit	Cluster Performance Monitors Common Event Identification register 3
0xE40	CLUSTERPMU_PMMIR	See individual bit resets.	32-bit	Cluster Performance Monitors Machine Identification Register
0xF00	CLUSTERPMU_PMITCTRL	See individual bit resets.	32-bit	Cluster Performance Monitors Integration mode Control register
0xFA0	CLUSTERPMU_PMCLAIMSET_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Claim Set register

Offset	Name	Reset	Width	Description
0xFA4	CLUSTERPMU_PMCLAIMCLR_EL1	See individual bit resets.	32-bit	Cluster Performance Monitors Claim Clear register
0xFA8	CLUSTERPMU_PMDEVAFF0	See individual bit resets.	32-bit	Cluster Performance Monitors Device Affinity register 0
0xFAC	CLUSTERPMU_PMDEVAFF1	See individual bit resets.	32-bit	Cluster Performance Monitors Device Affinity register 1
0xFB0	CLUSTERPMU_PMLAR	See individual bit resets.	32-bit	Cluster Performance Monitors Lock Access Register
0xFB4	CLUSTERPMU_PMLSR	See individual bit resets.	32-bit	Cluster Performance Monitors Lock Status Register
0xFB8	CLUSTERPMU_PMAUTHSTATUS	See individual bit resets.	32-bit	Cluster Performance Monitors Authentication Status register
0xFBC	CLUSTERPMU_PMDEVARCH	See individual bit resets.	32-bit	Cluster Performance Monitors Device Architecture register
0xFC8	CLUSTERPMU_PMDEVID	See individual bit resets.	32-bit	Cluster Performance Monitors Device ID register
0xFCC	CLUSTERPMU_PMDEVTYPE	See individual bit resets.	32-bit	Cluster Performance Monitors Device Type register
0xFD0	CLUSTERPMU_PMPIDR4	See individual bit resets.	32-bit	Cluster Performance Monitors Peripheral Identification Register 4
0xFE0	CLUSTERPMU_PMPIDR0	See individual bit resets.	32-bit	Cluster Performance Monitors Peripheral Identification Register 0
0xFE4	CLUSTERPMU_PMPIDR1	See individual bit resets.	32-bit	Cluster Performance Monitors Peripheral Identification Register 1
0xFE8	CLUSTERPMU_PMPIDR2	See individual bit resets.	32-bit	Cluster Performance Monitors Peripheral Identification Register 2
0xFEC	CLUSTERPMU_PMPIDR3	See individual bit resets.	32-bit	Cluster Performance Monitors Peripheral Identification Register 3
0xFF0	CLUSTERPMU_PMCIDR0	See individual bit resets.	32-bit	Cluster Performance Monitors Component Identification Register 0
0xFF4	CLUSTERPMU_PMCIDR1	See individual bit resets.	32-bit	Cluster Performance Monitors Component Identification Register 1
0xFF8	CLUSTERPMU_PMCIDR2	See individual bit resets.	32-bit	Cluster Performance Monitors Component Identification Register 2
0xFFC	CLUSTERPMU_PMCIDR3	See individual bit resets.	32-bit	Cluster Performance Monitors Component Identification Register 3

B.2.1.3 External DBROM registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped DBROM 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-216: DBROM registers summary

Offset	Name	Reset	Width	Description
0x000	DBROM_ROMENTRY0	See individual bit resets.	32-bit	DebugBlock ROM table Entry 0
0x004	DBROM_ROMENTRY1	See individual bit resets.	32-bit	DebugBlock ROM table Entry 1
0x008	DBROM_ROMENTRY2	See individual bit resets.	32-bit	DebugBlock ROM table Entry 2
0x00C	DBROM_ROMENTRY3	See individual bit resets.	32-bit	DebugBlock ROM table Entry 3
0x010	DBROM_ROMENTRY4	See individual bit resets.	32-bit	DebugBlock ROM table Entry 4
0x014	DBROM_ROMENTRY5	See individual bit resets.	32-bit	DebugBlock ROM table Entry 5
0x018	DBROM_ROMENTRY6	See individual bit resets.	32-bit	DebugBlock ROM table Entry 6
0x01C	DBROM_ROMENTRY7	See individual bit resets.	32-bit	DebugBlock ROM table Entry 7
0x020	DBROM_ROMENTRY8	See individual bit resets.	32-bit	DebugBlock ROM table Entry 8
0x024	DBROM_ROMENTRY9	See individual bit resets.	32-bit	DebugBlock ROM table Entry 9
0xA00	DBROM_DBGPCRO	See individual bit resets.	32-bit	DebugBlock ROM table Debug Power Control Register 0
0xA80	DBROM_DBGPSRO	See individual bit resets.	32-bit	DebugBlock ROM table Debug Power Status Register 0
0xC00	DBROM_PRIDRO	See individual bit resets.	32-bit	DebugBlock ROM table Power Request ID Register 0
0xF00	DBROM_ITCTRL	See individual bit resets.	32-bit	DebugBlock ROM table Integration Mode Control Register
0xFA0	DBROM_CLAIMSET	See individual bit resets.	32-bit	DebugBlock ROM table Claim Tag Set Register
0xFA4	DBROM_CLAIMCLR	See individual bit resets.	32-bit	DebugBlock ROM table Claim Tag Clear Register
0xFA8	DBROM_DEVAFF0	See individual bit resets.	32-bit	DebugBlock ROM table Device Affinity Register 0
0xFAC	DBROM_DEVAFF1	See individual bit resets.	32-bit	DebugBlock ROM table Device Affinity Register 1
0xFB0	DBROM_LAR	See individual bit resets.	32-bit	DebugBlock ROM table Software Lock Access Register
0xFB4	DBROM_LSR	See individual bit resets.	32-bit	DebugBlock ROM table Software Lock Status Register
0xFB8	DBROM_AUTHSTATUS	See individual bit resets.	32-bit	DebugBlock ROM table Authentication Status Register
0xFBC	DBROM_DEVARCH	See individual bit resets.	32-bit	DebugBlock ROM table Device Architecture Register
0xFC0	DBROM_DEVID2	See individual bit resets.	32-bit	DebugBlock ROM table Device Configuration Register 2
0xFC4	DBROM_DEVID1	See individual bit resets.	32-bit	DebugBlock ROM table Device Configuration Register 1
0xFC8	DBROM_DEVID	See individual bit resets.	32-bit	DebugBlock ROM table Device Configuration Register
0xFCC	DBROM_DEVTYPE	See individual bit resets.	32-bit	DebugBlock ROM table Device Type Register
0xFD0	DBROM_PIDR4	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 4
0xFD4	DBROM_PIDR5	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 5
0xFD8	DBROM_PIDR6	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 6
0xFDC	DBROM_PIDR7	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 7
0xFE0	DBROM_PIDR0	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 0
0xFE4	DBROM_PIDR1	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 1
0xFE8	DBROM_PIDR2	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 2
0xFEC	DBROM_PIDR3	See individual bit resets.	32-bit	DebugBlock ROM table Peripheral Identification Register 3
0xFF0	DBROM_CIDR0	See individual bit resets.	32-bit	DebugBlock ROM table Component Identification Register 0
0xFF4	DBROM_CIDR1	See individual bit resets.	32-bit	DebugBlock ROM table Component Identification Register 1
0xFF8	DBROM_CIDR2	See individual bit resets.	32-bit	DebugBlock ROM table Component Identification Register 2
0xFFC	DBROM_CIDR3	See individual bit resets.	32-bit	DebugBlock ROM table Component Identification Register 3

B.2.1.4 External CLUSTERROM registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped CLUSTERROM 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-217: CLUSTERROM registers summary

Offset	Name	Reset	Width	Description
0x000	CLUSTERROM_ROMENTRY0	See individual bit resets.	32-bit	Cluster ROM table Entry 0
0x004	CLUSTERROM_ROMENTRY1	See individual bit resets.	32-bit	Cluster ROM table Entry 1
0x008	CLUSTERROM_ROMENTRY2	See individual bit resets.	32-bit	Cluster ROM table Entry 2
0x00C	CLUSTERROM_ROMENTRY3	See individual bit resets.	32-bit	Cluster ROM table Entry 3
0x010	CLUSTERROM_ROMENTRY4	See individual bit resets.	32-bit	Cluster ROM table Entry 4
0x014	CLUSTERROM_ROMENTRY5	See individual bit resets.	32-bit	Cluster ROM table Entry 5
0x018	CLUSTERROM_ROMENTRY6	See individual bit resets.	32-bit	Cluster ROM table Entry 6
0x01C	CLUSTERROM_ROMENTRY7	See individual bit resets.	32-bit	Cluster ROM table Entry 7
0x020	CLUSTERROM_ROMENTRY8	See individual bit resets.	32-bit	Cluster ROM table Entry 8
0x024	CLUSTERROM_ROMENTRY9	See individual bit resets.	32-bit	Cluster ROM table Entry 9
0xA00	CLUSTERROM_DBGPCR0	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 0
0xA04	CLUSTERROM_DBGPCR1	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 1
0xA08	CLUSTERROM_DBGPCR2	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 2
0xA0C	CLUSTERROM_DBGPCR3	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 3
0xA10	CLUSTERROM_DBGPCR4	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 4
0xA14	CLUSTERROM_DBGPCR5	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 5
0xA18	CLUSTERROM_DBGPCR6	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 6
0xA1C	CLUSTERROM_DBGPCR7	See individual bit resets.	32-bit	Cluster ROM table Debug Power Control Register 7
0xA80	CLUSTERROM_DBGPSR0	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 0
0xA84	CLUSTERROM_DBGPSR1	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 1
0xA88	CLUSTERROM_DBGPSR2	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 2
0xA8C	CLUSTERROM_DBGPSR3	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 3
0xA90	CLUSTERROM_DBGPSR4	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 4
0xA94	CLUSTERROM_DBGPSR5	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 5
0xA98	CLUSTERROM_DBGPSR6	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 6
0xA9C	CLUSTERROM_DBGPSR7	See individual bit resets.	32-bit	Cluster ROM table Debug Power Status Register 7
0xC00	CLUSTERROM_PRIDR0	See individual bit resets.	32-bit	Cluster ROM table Power Request ID Register 0
0xF00	CLUSTERROM_ITCTRL	See individual bit resets.	32-bit	Cluster ROM table Integration Mode Control Register
0xFA0	CLUSTERROM_CLAIMSET	See individual bit resets.	32-bit	Cluster ROM table Claim Tag Set Register
0xFA4	CLUSTERROM_CLAIMCLR	See individual bit resets.	32-bit	Cluster ROM table Claim Tag Clear Register

Offset	Name	Reset	Width	Description
0xFA8	CLUSTERROM_DEVAFF0	See individual bit resets.	32-bit	Cluster ROM table Device Affinity Register 0
0xFAC	CLUSTERROM_DEVAFF1	See individual bit resets.	32-bit	Cluster ROM table Device Affinity Register 1
0xFB0	CLUSTERROM_LAR	See individual bit resets.	32-bit	Cluster ROM table Software Lock Access Register
0xFB4	CLUSTERROM_LSR	See individual bit resets.	32-bit	Cluster ROM table Software Lock Status Register
0xFB8	CLUSTERROM_AUTHSTATUS	See individual bit resets.	32-bit	Cluster ROM table Authentication Status Register
0xFBC	CLUSTERROM_DEVARCH	See individual bit resets.	32-bit	Cluster ROM table Device Architecture Register
0xFC0	CLUSTERROM_DEVID2	See individual bit resets.	32-bit	Cluster ROM table Device Configuration Register 2
0xFC4	CLUSTERROM_DEVID1	See individual bit resets.	32-bit	Cluster ROM table Device Configuration Register 1
0xFC8	CLUSTERROM_DEVID	See individual bit resets.	32-bit	Cluster ROM table Device Configuration Register
0xFCC	CLUSTERROM_DEVTYPE	See individual bit resets.	32-bit	Cluster ROM table Device Type Register
0xFD0	CLUSTERROM_PIDR4	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 4
0xFD4	CLUSTERROM_PIDR5	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 5
0xFD8	CLUSTERROM_PIDR6	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 6
0xFDC	CLUSTERROM_PIDR7	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 7
0xFE0	CLUSTERROM_PIDR0	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 0
0xFE4	CLUSTERROM_PIDR1	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 1
0xFE8	CLUSTERROM_PIDR2	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 2
0xFEC	CLUSTERROM_PIDR3	See individual bit resets.	32-bit	Cluster ROM table Peripheral Identification Register 3
0xFF0	CLUSTERROM_CIDR0	See individual bit resets.	32-bit	Cluster ROM table Component Identification Register 0
0xFF4	CLUSTERROM_CIDR1	See individual bit resets.	32-bit	Cluster ROM table Component Identification Register 1
0xFF8	CLUSTERROM_CIDR2	See individual bit resets.	32-bit	Cluster ROM table Component Identification Register 2
0xFFC	CLUSTERROM_CIDR3	See individual bit resets.	32-bit	Cluster ROM table Component Identification Register 3

B.2.1.5 External COREROM registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-218: 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
0xC00	COREROM_PRIDR0	See individual bit resets.	32-bit	Core ROM table Power Request ID Register 0

Offset	Name	Reset	Width	Description
0xF00	COREROM_ITCTRL	See individual bit resets.	32-bit	Core ROM table Integration Mode Control Register
0xFA0	COREROM_CLAIMSET	See individual bit resets.	32-bit	Core ROM table Claim Tag Set Register
0xFA4	COREROM_CLAIMCLR	See individual bit resets.	32-bit	Core ROM table Claim Tag Clear Register
0xFA8	COREROM_DEVAFF0	See individual bit resets.	32-bit	Core ROM table Device Affinity Register 0
0xFAC	COREROM_DEVAFF1	See individual bit resets.	32-bit	Core ROM table Device Affinity Register 1
0xFB0	COREROM_LAR	See individual bit resets.	32-bit	Core ROM table Software Lock Access Register
0xFB4	COREROM_LSR	See individual bit resets.	32-bit	Core ROM table Software Lock Status Register
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
0xFC0	COREROM_DEVID2	See individual bit resets.	32-bit	Core ROM table Device Configuration Register 2
0xFC4	COREROM_DEVID1	See individual bit resets.	32-bit	Core ROM table Device Configuration Register 1
0xFC8	COREROM_DEVID	See individual bit resets.	32-bit	Core ROM table Device Configuration 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
0xFD4	COREROM_PIDR5	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 5
0xFD8	COREROM_PIDR6	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 6
0xFDC	COREROM_PIDR7	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 7
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.2.1.6 External CTI registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-219: CTI registers summary

Offset	Name	Reset	Width	Description
0x000	CTICONTROL	See individual bit resets.	32-bit	CTI Control register
0x010	CTIINTACK	See individual bit resets.	32-bit	CTI Output Trigger Acknowledge register

Offset	Name	Reset	Width	Description
0x014	CTIAPPSET	See individual bit resets.	32-bit	CTI Application Trigger Set register
0x018	CTIAPPCLEAR	See individual bit resets.	32-bit	CTI Application Trigger Clear register
0x01C	CTIAPPULSE	See individual bit resets.	32-bit	CTI Application Pulse register
0x020 + (4 * n)	CTIINEN<n>	See individual bit resets.	32-bit	CTI Input Trigger to Output Channel Enable registers
0x0A0 + (4 * n)	CTIOUTEN<n>	See individual bit resets.	32-bit	CTI Input Channel to Output Trigger Enable registers
0x130	CTITRIGINSTATUS	See individual bit resets.	32-bit	CTI Trigger In Status register
0x134	CTITRIGOUTSTATUS	See individual bit resets.	32-bit	CTI Trigger Out Status register
0x138	CTICHINSTATUS	See individual bit resets.	32-bit	CTI Channel In Status register
0x13C	CTICHOUTSTATUS	See individual bit resets.	32-bit	CTI Channel Out Status register
0x140	CTIGATE	See individual bit resets.	32-bit	CTI Channel Gate Enable register
0x144	ASICCTL	See individual bit resets.	32-bit	CTI External Multiplexer Control register
0x150	CTIDEVCTL	See individual bit resets.	32-bit	CTI Device Control register
0xF00	CTIITCTRL	See individual bit resets.	32-bit	CTI Integration mode Control register
0xFA0	CTICLAIMSET	See individual bit resets.	32-bit	CTI Claim Tag Set register
0xFA4	CTICLAIMCLR	See individual bit resets.	32-bit	CTI Claim Tag Clear register
0xFA8	CTIDEVAFF0	See individual bit resets.	32-bit	CTI Device Affinity register 0
0xFAC	CTIDEVAFF1	See individual bit resets.	32-bit	CTI Device Affinity register 1
0xFB0	CTILAR	See individual bit resets.	32-bit	CTI Lock Access Register
0xFB4	CTILSR	See individual bit resets.	32-bit	CTI Lock Status Register
0xFB8	CTIAUTHSTATUS	See individual bit resets.	32-bit	CTI Authentication Status register
0xFBC	CTIDEVARCH	See individual bit resets.	32-bit	CTI Device Architecture register
0xFC0	CTIDEVID2	See individual bit resets.	32-bit	CTI Device ID register 2
0xFC4	CTIDEVID1	See individual bit resets.	32-bit	CTI Device ID register 1
0xFC8	CTIDEVID	See individual bit resets.	32-bit	CTI Device ID register 0
0xFCC	CTIDEVTYPE	See individual bit resets.	32-bit	CTI Device Type register
0xFD0	CTIPIDR4	See individual bit resets.	32-bit	CTI Peripheral Identification Register 4
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
0xFE8	CTIPIDR2	See individual bit resets.	32-bit	CTI Peripheral Identification Register 2
0xFEC	CTIPIDR3	See individual bit resets.	32-bit	CTI Peripheral Identification Register 3
0xFF0	CTICIDR0	See individual bit resets.	32-bit	CTI Component Identification Register 0
0xFF4	CTICIDR1	See individual bit resets.	32-bit	CTI Component Identification Register 1
0xFF8	CTICIDR2	See individual bit resets.	32-bit	CTI Component Identification Register 2
0xFFC	CTICIDR3	See individual bit resets.	32-bit	CTI Component Identification Register 3

B.2.1.7 External ETM registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** memory-mapped ETM 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 [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-220: ETM registers summary

Offset	Name	Reset	Width	Description
0x004	TRCPRGCTLR	See individual bit resets.	32-bit	Programming Control Register
0x00C	TRCSTATR	See individual bit resets.	32-bit	Trace Status Register
0x010	TRCCONFIGR	See individual bit resets.	32-bit	Trace Configuration Register
0x018	TRCAUXCTLR	See individual bit resets.	32-bit	Auxillary Control Register
0x020	TRCEVENTCTLOR	See individual bit resets.	32-bit	Event Control 0 Register
0x024	TRCEVENTCTL1R	See individual bit resets.	32-bit	Event Control 1 Register
0x02C	TRCSTALLCTLR	See individual bit resets.	32-bit	Stall Control Register
0x030	TRCTSCTLR	See individual bit resets.	32-bit	Timestamp Control Register
0x034	TRCSYNCPR	See individual bit resets.	32-bit	Synchronization Period Register
0x038	TRCCCCTLR	See individual bit resets.	32-bit	Cycle Count Control Register
0x03C	TRCBBCTLR	See individual bit resets.	32-bit	Branch Broadcast Control Register
0x040	TRCTRACEIDR	See individual bit resets.	32-bit	Trace ID Register
0x080	TRCVICTLR	See individual bit resets.	32-bit	ViewInst Main Control Register
0x084	TRCVIIECTLR	See individual bit resets.	32-bit	ViewInst Include/Exclude Control Register
0x088	TRCVISSCTLR	See individual bit resets.	32-bit	ViewInst Start/Stop Control Register
0x0A0	TRCVDCTLR	See individual bit resets.	32-bit	ViewData Main Control Register
0x0A4	TRCVDSACCTLR	See individual bit resets.	32-bit	ViewData Include/Exclude Single Address Comparator Control Register

Offset	Name	Reset	Width	Description
0x0A8	TRCVDARCTL	See individual bit resets.	32-bit	ViewData Include/Exclude Address Range Comparator Control Register
0x100 + (4 * n)	TRCSEQEVR<n>	See individual bit resets.	32-bit	Sequencer State Transition Control Register <n>
0x118	TRCSEQRSTEVR	See individual bit resets.	32-bit	Sequencer Reset Control Register
0x11C	TRCSEQSTR	See individual bit resets.	32-bit	Sequencer State Register
0x120	TRCEXTINSEL	See individual bit resets.	32-bit	External Input Select Register
0x140 + (4 * n)	TRCCNTRLDVR<n>	See individual bit resets.	32-bit	Counter Reload Value Register <n>
0x150 + (4 * n)	TRCCNTCTL<n>	See individual bit resets.	32-bit	Counter Control Register <n>
0x160 + (4 * n)	TRCCNTVR<n>	See individual bit resets.	32-bit	Counter Value Register <n>
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	TRCIMSPECO	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

Offset	Name	Reset	Width	Description
0x200 + (4 * n)	TRCRSCTLR<n>	See individual bit resets.	32-bit	Resource Selection Control Register <n>
0x280 + (4 * n)	TRCSSCCR<n>	See individual bit resets.	32-bit	Single-shot Comparator Control Register <n>
0x2A0 + (4 * n)	TRCSSCSR<n>	See individual bit resets.	32-bit	Single-shot Comparator Control Status Register <n>
0x300	TRCOSLAR	See individual bit resets.	32-bit	Trace OS Lock Access Register
0x304	TRCOSLSR	See individual bit resets.	32-bit	Trace OS Lock Status Register
0x310	TRCPDCR	See individual bit resets.	32-bit	PowerDown Control Register
0x314	TRCPDSR	See individual bit resets.	32-bit	PowerDown Status Register
0x400 + (8 * n)	TRCACVR<n>	See individual bit resets.	64-bit	Address Comparator Value Register <n>
0x480 + (8 * n)	TRCACATR<n>	See individual bit resets.	64-bit	Address Comparator Access Type Register <n>
0x500 + (16 * n)	TRCDVCVR<n>	See individual bit resets.	64-bit	Data Value Comparator Value Register <n>
0x580 + (16 * n)	TRCDVCMR<n>	See individual bit resets.	64-bit	Data Value Comparator Mask Register <n>
0x600 + (8 * n)	TRCCIDCVR<n>	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>
0x640 + (8 * n)	TRCVMIDCVR<n>	See individual bit resets.	64-bit	Virtual Context Identifier Comparator Value Register <n>
0x680	TRCCIDCCTLRO	See individual bit resets.	32-bit	Context Identifier Comparator Control Register 0
0x688	TRCVMIDCCTLRO	See individual bit resets.	32-bit	Virtual Context Identifier Comparator Control Register 0
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
0xFA8	TRCDEVAFF	See individual bit resets.	64-bit	Device Affinity Register
0xFB0	TRCLAR	See individual bit resets.	32-bit	Lock Access Register
0xFB4	TRCLSR	See individual bit resets.	32-bit	Lock Status Register
0xFB8	TRCAUTHSTATUS	See individual bit resets.	32-bit	Authentication Status Register
0xFBC	TRCDEVARCH	See individual bit resets.	32-bit	Device Architecture Register

Offset	Name	Reset	Width	Description
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.2.1.8 External PMU registers summary

The following summary table provides an overview of **IMPLEMENTATION DEFINED** 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.

For registers without a listed reset value, refer to the individual field resets documented on the register description pages or to the [Arm® Architecture Reference Manual for A-profile architecture](#).

Table B-221: PMU registers summary

Offset	Name	Reset	Width	Description
0x000 + (8 * n)	PMEVCNTR<n>_ELO [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_ELO [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter

Offset	Name	Reset	Width	Description
0x0FC	PMCCNTR_ELO [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
0x20C	PMVIDSR	See individual bit resets.	32-bit	VMID Sample Register
0x22C	PMCID2SR	See individual bit resets.	32-bit	CONTEXTIDR_EL2 Sample Register
0x400 + (4 * n)	PMEVTYPER<n>_ELO [31:0]	See individual bit resets.	32-bit	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_ELO [31:0]	See individual bit resets.	32-bit	Performance Monitors Cycle Counter Filter Register
0xC00	PMCNTENSET_ELO [31:0]	See individual bit resets.	32-bit	Performance Monitors Count Enable Set Register
0xC20	PMCNTENCLR_ELO [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_ELO [31:0]	See individual bit resets.	32-bit	Performance Monitors Overflow Flag Status Clear register
0xCA0	PMSWINC_ELO	See individual bit resets.	32-bit	Performance Monitors Software Increment Register
0xCC0	PMOVSSET_ELO [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_ELO	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

Offset	Name	Reset	Width	Description
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE40	PMMIR [31:0]	See individual bit resets.	32-bit	Performance Monitors Machine Identification Register
0xF00	PMITCTRL	See individual bit resets.	32-bit	Performance Monitors Integration mode Control 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.2.2 Register descriptions for registers accessed over the Debug APB bus

This section includes the descriptions for all the external registers in the Cortex®-R82 processor accessed over the Debug APB bus.

B.2.2.1 External Debug register description

This section includes the register descriptions for all memory-mapped Debug registers that are accessed for each core.

B.2.2.1.1 EDESR, External Debug Event Status Register

Indicates the status of internally pending Halting debug events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

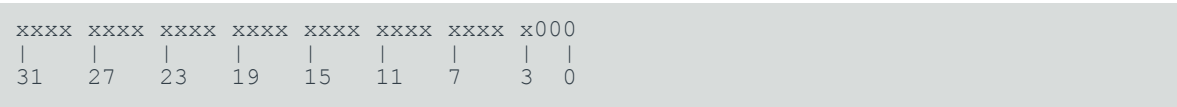
Register offset

0x020

Access type

RW1C

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-107: EXT_EDESR bit assignments

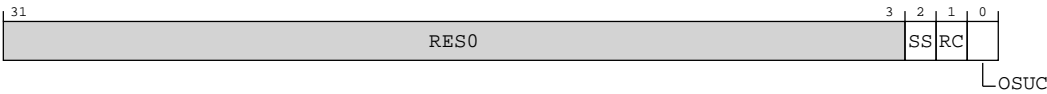


Table B-222: EDESR bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2]	SS	<p>Halting step debug event pending. Possible values of this field are:</p> <p>0b0</p> <p>Reading this means that a Halting step debug event is not pending. Writing this means no action.</p> <p>0b1</p> <p>Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.</p>	0b0
[1]	RC	<p>Reset Catch debug event pending. Possible values of this field are:</p> <p>0b0</p> <p>Reading this means that a Reset Catch debug event is not pending. Writing this means no action.</p> <p>0b1</p> <p>Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.</p>	0b0
[0]	OSUC	<p>OS Unlock Catch debug event pending. Possible values of this field are:</p> <p>0b0</p> <p>Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action.</p> <p>0b1</p> <p>Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.</p>	0b0

Access

If a request to clear a pending Halting debug event is received at or about the time when halting becomes allowed, it is **CONSTRAINED UNPREDICTABLE** whether the event is taken.

If Core power is removed while a Halting debug event is pending, it is lost. However, it might become pending again when the Core is powered back on and Cold reset.

Accessibility

If a request to clear a pending Halting debug event is received at or about the time when halting becomes allowed, it is **CONSTRAINED UNPREDICTABLE** whether the event is taken.

If Core power is removed while a Halting debug event is pending, it is lost. However, it might become pending again when the Core is powered back on and Cold reset.

Component	Offset	Instance	Range
Debug	0x020	EDESR	None

This interface is accessible as follows:

When IsCorePowered()

RW

Otherwise

ERROR

B.2.2.1.2 EDECR, External Debug Execution Control Register

Controls Halting debug events.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

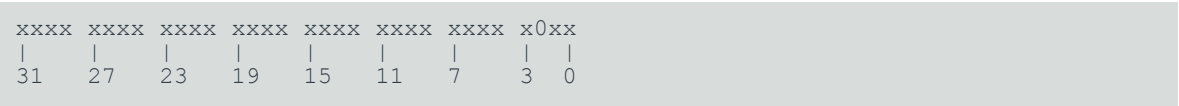
Register offset

0x024

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-108: EXT_EDECR bit assignments

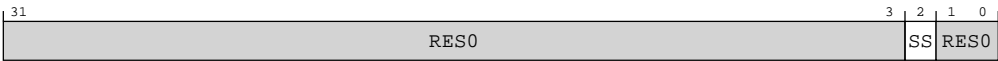


Table B-224: EDECR bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2]	SS	Halting step enable. Possible values of this field are: 0b0 Halting step debug event disabled. 0b1 Halting step debug event enabled. If the value of EDECR.SS is changed when the PE is in Non-debug state, behavior is CONstrained UNPREDICTABLE. The implemented behavior is the value of ext-EDECR.SS becomes UNKNOWN .	0b0
[1:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
Debug	0x024	EDECR	None

This interface is accessible as follows:

When IsCorePowered()

RW

Otherwise

ERROR

B.2.2.1.3 EDWAR, External Debug Watchpoint Address Register

Returns the virtual data address being accessed when a Watchpoint Debug Event was triggered.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0x030

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-109: EXT_EDWAR bit assignments

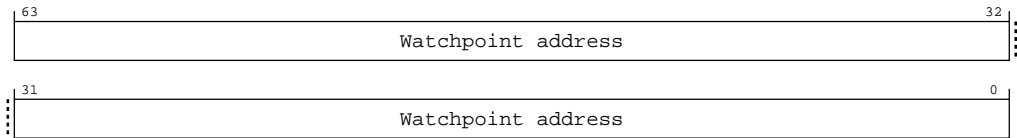


Table B-226: EDWAR bit descriptions

Bits	Name	Description	Reset
[63:0]	None	<p>Watchpoint address. The data virtual address being accessed when a Watchpoint Debug Event was triggered and caused entry to Debug state. This address must be within a naturally-aligned block of memory of power-of-two size no larger than the DC ZVA block size.</p> <p>The value of this register is UNKNOWN if the PE is in Non-debug state, or if Debug state was entered other than for a Watchpoint debug event.</p> <p>The EDWAR is subject to the same alignment rules as the reporting of a watchpointed address in the FAR. See <i>Determining the memory location that caused a Watchpoint exception</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	64 {x}

Accessibility

Component	Offset	Instance	Range
Debug	0x030	EDWAR	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RO

Otherwise

ERROR

B.2.2.1.4 DBGDTRRX_EL0, Debug Data Transfer Register, Receive

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. See AArch64-DBGDTR_EL0 for additional architectural mappings. It is a component of the Debug Communications Channel.

Configurations

External register DBGDTRRX_EL0 bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.20 DBGDTRRX_EL0, Debug Data Transfer Register, Receive](#) on page 1071 bits [31:0].

Attributes

Width

32

Component

Debug

Register offset

0x080

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-110: EXT_DBGDTRRX_EL0 bit assignments

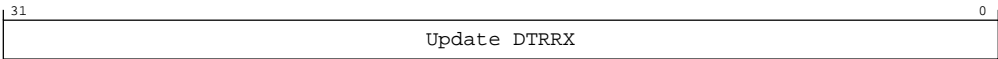


Table B-228: DBGDTRRX_ELO bit descriptions

Bits	Name	Description	Reset
[31:0]	None	<p>Update DTRRX.</p> <p>Writes to this register:</p> <ul style="list-style-type: none"> If RXfull is set to 1, set DTRRX to UNKNOWN. If RXfull is set to 0, update the value in DTRRX. <p>After the write, RXfull is set to 1.</p> <p>Reads of this register:</p> <ul style="list-style-type: none"> If RXfull is set to 1, return the last value written to DTRRX. If RXfull is set to 0, return an UNKNOWN value. <p>After the read, RXfull remains unchanged.</p> <p>For the full behavior of the Debug Communications Channel, see <i>The Debug Communication Channel and Instruction Transfer Register</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	32 {x}

Access

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

Accessibility

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

Component	Offset	Instance	Range
Debug	0x080	DBGDTRRX_ELO	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ERROR

B.2.2.1.5 EDITR, External Debug Instruction Transfer Register

Used in Debug state for passing instructions to the PE for execution.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

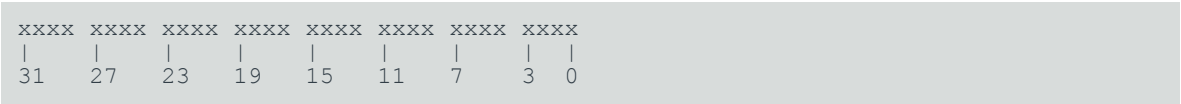
Register offset

0x084

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-111: EXT_EDITR bit assignments

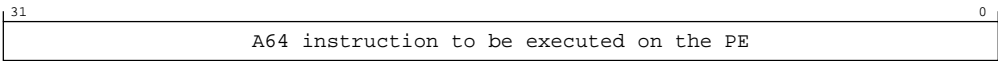


Table B-230: EDITR bit descriptions

Bits	Name	Description	Reset
[31:0]	None	A64 instruction to be executed on the PE.	32 {x}

Access

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any instruction issued through the ITR in Normal access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.

- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

EDITR ignores writes if the PE is in Non-debug state.

Accessibility

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any instruction issued through the ITR in Normal access mode that has not completed execution is CONSTRAINED UNPREDICTABLE, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an UNKNOWN state.

EDITR ignores writes if the PE is in Non-debug state.

Component	Offset	Instance	Range
Debug	0x084	EDITR	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()
WO
Otherwise
ERROR

B.2.2.1.6 EDSCR, External Debug Status and Control Register

Main control register for the debug implementation.

Configurations

External register EDSCR bits [30:29] are architecturally mapped to AArch64 System register [A.2.5.18 MDCCSR_ELO, Monitor DCC Status Register](#) on page 1067 bits [30:29].

Attributes

Width
32
Component
Debug
Register offset
0x088
Access type
inconsistent

Reset value

000x	00xx	0000	xxxx	x011	11xx	x0xx	xxxx
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-112: EXT_EDSCR bit assignments

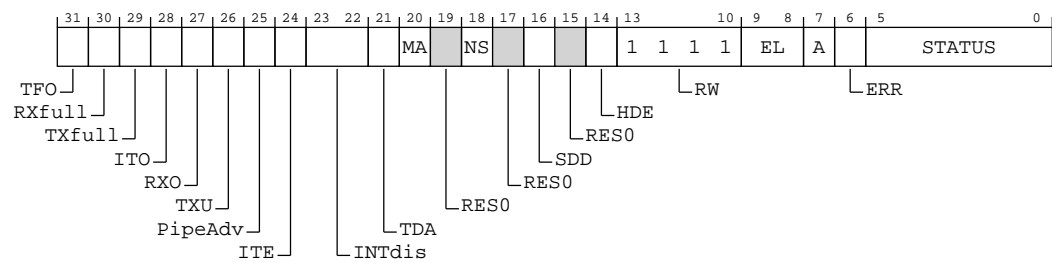


Table B-232: EDSCR bit descriptions

Bits	Name	Description	Reset
[31]	TFO	Trace Filter Override. Overrides the Trace Filter controls allowing the external debugger to trace any visible Exception level. 0b0 Trace Filter controls are not affected. 0b1 Trace Filter controls in AArch64-TRFCR_EL1 and AArch64-TRFCR_EL2 are ignored. When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers: <ul style="list-style-type: none">AArch64-MDSCR_EL1. This bit is ignored by the PE when ExternalSecureNoninvasiveDebugEnabled() is FALSE and the Effective value of AArch64-MDCR_EL3.STE is 1.	0b0
[30]	RXfull	DTRRX full. When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers: <ul style="list-style-type: none">AArch64-MDSCR_EL1. Access to this field is: RO	0b0

Bits	Name	Description	Reset
[29]	TXfull	DTRTX full. When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers: <ul style="list-style-type: none"> AArch64-MDSCR_EL1. Access to this field is: RO	0b0
[28]	ITO	ITR overrun. Set to 0 on entry to Debug state. When !Halted() Access to this field is: UNKNOWN/WI Otherwise Access to this field is: RO	x
[27]	RXO	DTRRX overrun. When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers: <ul style="list-style-type: none"> AArch64-MDSCR_EL1. Access to this field is: RO	0b0
[26]	TXU	DTRTX underrun. When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers: <ul style="list-style-type: none"> AArch64-MDSCR_EL1. Access to this field is: RO	0b0
[25]	PipeAdv	Pipeline Advance. Indicates that software execution is progressing. 0b0 No progress has been made by the PE since the last time this field was cleared to zero by writing 1 to ext-EDRCR.CSPA. 0b1 Progress has been made by the PE since the last time this field was cleared to zero by writing 1 to ext-EDRCR.CSPA. The architecture does not define precisely when this field is set to 1. It requires only that this happen periodically in Non-debug state to indicate that software execution is progressing. For example, a PE might set this field to 1 each time the PE retires one or more instructions, or at periodic intervals during the progression of an instruction. Access to this field is: RO	x
[24]	ITE	ITR empty. When !Halted() Access to this field is: UNKNOWN/WI Otherwise Access to this field is: RO	x

Bits	Name	Description	Reset
[23:22]	INTdis	<p>Interrupt disable. Disables taking interrupts in Non-debug state.</p> <p>0b00</p> <p>Masking of interrupts is controlled by PSTATE and interrupt routing controls.</p> <p>0b01</p> <p>If ExternalInvasiveDebugEnabled() is TRUE, then all interrupts taken to Non-secure state are masked.</p> <p>If ExternalSecureInvasiveDebugEnabled() is TRUE, then all interrupts taken to Secure state are masked.</p> <p>Note:</p> <p>All interrupts includes virtual and SError interrupts.</p> <p>When AArch64-OSLSR_EL1.OSLK is 1, this field can be indirectly read and written through the following System registers:</p> <ul style="list-style-type: none"> AArch64-MDSCR_EL1. <p>The Effective value of this field is 0b00 when ExternalInvasiveDebugEnabled() is FALSE.</p> <p>When FEAT_Debugv8p4 is implemented, bit[23] of this register is RES0.</p>	0b00
[21]	TDA	<p>Traps accesses to the following debug System registers:</p> <ul style="list-style-type: none"> AArch64: AArch64-DBGBCR<n>_EL1, AArch64-DBGBVR<n>_EL1, AArch64-DBGWCR<n>_EL1, AArch64-DBGWVR<n>_EL1. <p>0b0</p> <p>Accesses to debug System registers do not generate a Software Access Debug event.</p> <p>0b1</p> <p>Accesses to debug System registers generate a Software Access Debug event, if AArch64-OSLSR_EL1.OSLK is 0 and if halting is allowed.</p> <p>When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers:</p> <ul style="list-style-type: none"> AArch64-MDSCR_EL1. 	0b0
[20]	MA	<p>Memory access mode. Controls the use of memory-access mode for accessing ITR and the DCC. This bit is ignored if in Non-debug state and set to zero on entry to Debug state.</p> <p>Possible values of this field are:</p> <p>0b0</p> <p>Normal access mode.</p> <p>0b1</p> <p>Memory access mode.</p>	0b0
[19]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[18]	NS	<p>Non-secure status. In Debug state, gives the current Security state:</p> <p>0b0 Secure state.</p> <p>0b1 Non-secure state.</p> <p>When !Halted() Access to this field is: UNKNOWN/WI</p> <p>Otherwise Access to this field is: RO</p>	x
[17]	RES0	Reserved	RES0
[16]	SDD	<p>Secure debug disabled.</p> <p>On entry to Debug state:</p> <ul style="list-style-type: none"> If entering in Secure state, SDD is set to 0. If entering in Non-secure state, SDD is set to the inverse of <code>ExternalSecureInvasiveDebugEnabled()</code>. <p>In Debug state, the value of the SDD bit does not change, even if <code>ExternalSecureInvasiveDebugEnabled()</code> changes.</p> <p>In Non-debug state:</p> <ul style="list-style-type: none"> SDD returns the inverse of <code>ExternalSecureInvasiveDebugEnabled()</code>. If the authentication signals that control <code>ExternalSecureInvasiveDebugEnabled()</code> change, a context synchronization event is required to guarantee their effect. This bit is unaffected by the Security state of the PE. <p>Access to this field is: RO</p>	x
[15]	RES0	Reserved	RES0
[14]	HDE	<p>Halting debug enable.</p> <p>0b0 Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events.</p> <p>0b1 Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.</p> <p>When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers:</p> <ul style="list-style-type: none"> AArch64-MDSCR_EL1. 	0b0

Bits	Name	Description	Reset
[13:10]	RW	<p>Exception level Execution state status. In Debug state, each bit gives the current Execution state of each Exception level.</p> <p>0b1111</p> <p>Any of the following:</p> <ul style="list-style-type: none"> The PE is in Non-debug state. The PE is at EL0 using AArch64. The PE is not at EL0, and both EL1 and EL2 are using AArch64. <p>When !Halted()</p> <p>Access to this field is: RAO/WI</p> <p>Otherwise</p> <p>Access to this field is: RO</p>	0b1111
[9:8]	EL	<p>Exception level. In Debug state, gives the current Exception level of the PE.</p> <p>When !Halted()</p> <p>Access to this field is: RAZ/WI</p> <p>Otherwise</p> <p>Access to this field is: RO</p>	xx
[7]	A	<p>SError interrupt pending. In Debug state, indicates whether an SError interrupt is pending:</p> <ul style="list-style-type: none"> If AArch64-HCR_EL2.{AMO, TGE} = {1, 0}, EL2 is enabled in the current Security state, and the PE is executing at EL0 or EL1, a virtual SError interrupt. Otherwise, a physical SError interrupt. <p>0b0</p> <p>No SError interrupt pending.</p> <p>0b1</p> <p>SError interrupt pending.</p> <p>A debugger can read EDSCR to check whether an SError interrupt is pending without having to execute further instructions. A pending SError might indicate data from target memory is corrupted.</p> <p>When !Halted()</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: RO</p>	x
[6]	ERR	<p>Cumulative error flag. This bit is set to 1 following exceptions in Debug state and on any signaled overrun or underrun on the DTR or EDITR.</p> <p>When AArch64-OSLSR_EL1.OSLK is 1, this bit can be indirectly read and written through the following System registers:</p> <ul style="list-style-type: none"> AArch64-MDSCR_EL1. <p>Access to this field is: RO</p>	0b0

Bits	Name	Description	Reset
[5:0]	STATUS	<p>Debug status flags.</p> <p>0b000001 PE is restarting, exiting Debug state.</p> <p>0b000010 PE is in Non-debug state.</p> <p>0b000111 Breakpoint.</p> <p>0b010011 External debug request.</p> <p>0b011011 Halting step, normal.</p> <p>0b011111 Halting step, exclusive.</p> <p>0b100011 OS Unlock Catch.</p> <p>0b100111 Reset Catch.</p> <p>0b101011 Watchpoint.</p> <p>0b101111 HLT instruction.</p> <p>0b110011 Software access to debug register.</p> <p>0b110111 Exception Catch.</p> <p>0b111011 Halting step, no syndrome.</p> <p>All other values of STATUS are reserved.</p> <p>Access to this field is: RO</p>	6{x}

Accessibility

Component	Offset	Instance	Range
Debug	0x088	EDSCR	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ERROR

B.2.2.1.7 DBGDTRTX_EL0, Debug Data Transfer Register, Transmit

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. See AArch64-DBGDTR_EL0 for additional architectural mappings. It is a component of the Debug Communication Channel.

Configurations

External register DBGDTRTX_EL0 bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.21 DBGDTRTX_EL0, Debug Data Transfer Register, Transmit](#) on page 1073 bits [31:0].

Attributes

Width

32

Component

Debug

Register offset

0x08C

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-113: EXT_DBGDTRTX_EL0 bit assignments

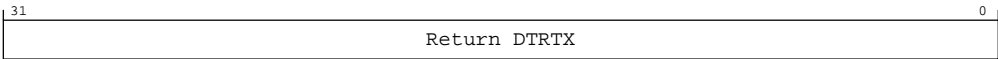


Table B-234: DBGDTRTX_ELO bit descriptions

Bits	Name	Description	Reset
[31:0]	None	<p>Return DTRTX.</p> <p>Reads of this register:</p> <ul style="list-style-type: none"> If TXfull is set to 1, return the last value written to DTRTX. If TXfull is set to 0, return an UNKNOWN value. <p>After the read, TXfull is cleared to 0.</p> <p>Writes to this register:</p> <ul style="list-style-type: none"> If TXfull is set to 1, set DTRTX to UNKNOWN. If TXfull is set to 0, update the value in DTRTX. <p>After the write, TXfull remains unchanged.</p> <p>For the full behavior of the Debug Communications Channel, see <i>The Debug Communication Channel and Instruction Transfer Register</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	32 {x}

Access

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

Accessibility

If ext-EDSCR.ITE == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an **UNKNOWN** state.

Component	Offset	Instance	Range
Debug	0x08C	DBGDTRTX_ELO	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ERROR

B.2.2.1.8 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in ext-EDSCR.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

Debug

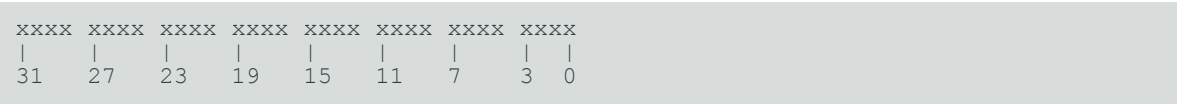
Register offset

0x090

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-114: EXT_EDRCR bit assignments

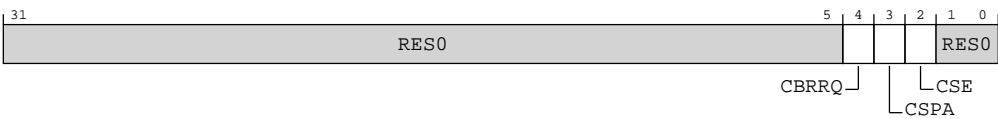


Table B-236: EDRCR bit descriptions

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[4]	CBRRQ	<p>Allow imprecise entry to Debug state. The actions on writing to this bit are:</p> <p>0b0 No action.</p> <p>0b1 Allow imprecise entry to Debug state, for example by canceling pending bus accesses.</p> <p>Setting this bit to 1 allows a debugger to request imprecise entry to Debug state. An External Debug Request debug event must be pending before the debugger sets this bit to 1.</p>	x
[3]	CSPA	<p>Clear Sticky Pipeline Advance. This bit is used to clear the ext-EDSCR.PipeAdv bit to 0. The actions on writing to this bit are:</p> <p>0b0 No action.</p> <p>0b1 Clear the ext-EDSCR.PipeAdv bit to 0.</p>	x
[2]	CSE	<p>Clear Sticky Error. Used to clear the ext-EDSCR cumulative error bits to 0. The actions on writing to this bit are:</p> <p>0b0 No action.</p> <p>0b1 Clear the ext-EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the ext-EDSCR.ITO bit, to 0.</p>	x
[1:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
Debug	0x090	EDRCR	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

WO

Otherwise

ERROR

B.2.2.1.9 EDECCR, External Debug Exception Catch Control Register

Controls Exception Catch debug events. For more information, see *Exception Catch debug event* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

External register EDECCR bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.5 OSECCR_EL1, OS Lock Exception Catch Control Register](#) on page 1036 bits [31:0].

Attributes

Width

32

Component

Debug

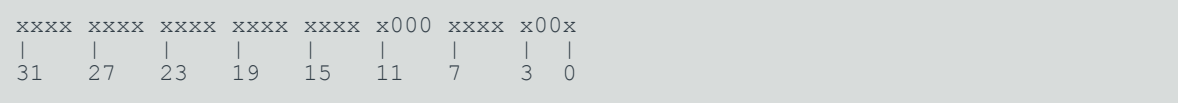
Register offset

0x098

Access type

RESORESO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-115: EXT_EDECCR bit assignments

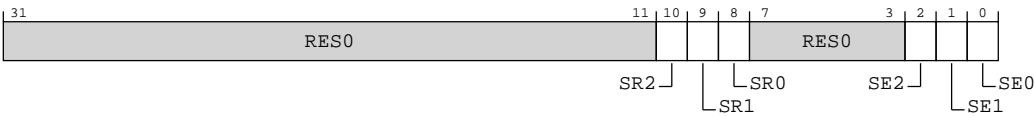


Table B-238: EDECCR bit descriptions

Bits	Name	Description	Reset
[31:11]	RES0	Reserved	RES0
[10]	SR2	Controls exception catch on exception return to Secure EL2 in conjunction with EDECCR.SE2. 0b0 If EDECCR.SE2 is 0, then Exception Catch debug events are disabled for Secure EL2. If EDECCR.SE2 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL2. 0b1 If EDECCR.SE2 is 0, then Exception Catch debug events are enabled for exception returns to Secure EL2. If EDECCR.SE2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL2.	0b0

Bits	Name	Description	Reset
[9]	SR1	<p>Controls exception catch on exception return to Secure EL1 in conjunction with EDECCR.SE1.</p> <p>0b0</p> <p>If EDECCR.SE1 is 0, then Exception Catch debug events are disabled for Secure EL1.</p> <p>If EDECCR.SE1 is 1, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL1.</p> <p>0b1</p> <p>If EDECCR.SE1 is 0, then Exception Catch debug events are enabled for exception returns to Secure EL1.</p> <p>If EDECCR.SE1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL1.</p>	0b0
[8]	SRO	<p>Controls exception catch on exception return to Secure EL0.</p> <p>0b0</p> <p>Exception Catch debug events are disabled for Secure EL0.</p> <p>0b1</p> <p>Exception Catch debug events are enabled for exception returns to Secure EL0.</p>	0b0
[7:3]	RES0	Reserved	RES0
[2]	SE2	<p>Controls exception catch on exception entry to Secure EL2. Also controls exception catch on exception return to Secure EL2 in conjunction with EDECCR.SR2.</p> <p>0b0</p> <p>If EDECCR.SR2 is 0, then Exception Catch debug events are disabled for Secure EL2.</p> <p>If EDECCR.SR2 is 1, then Exception Catch debug events are enabled for exception returns to Secure EL2.</p> <p>0b1</p> <p>If EDECCR.SR2 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL2.</p> <p>If EDECCR.SR2 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL2.</p> <p>Note:</p> <p>It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level will generate an Exception Catch debug event. In this implementation, a reset entry will do so.</p>	0b0

Bits	Name	Description	Reset
[1]	SE1	<p>Controls exception catch on exception entry to Secure EL1. Also controls exception catch on exception return to Secure EL1 in conjunction with EDECCR.SR1.</p> <p>0b0</p> <p>If EDECCR.SR1 is 0, then Exception Catch debug events are disabled for Secure EL1.</p> <p>If EDECCR.SR1 is 1, then Exception Catch debug events are enabled for exception returns to Secure EL1.</p> <p>0b1</p> <p>If EDECCR.SR1 is 0, then Exception Catch debug events are enabled for exception entry, reset entry, and exception return to Secure EL1.</p> <p>If EDECCR.SR1 is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure EL1.</p> <p>Note: It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level will generate an Exception Catch debug event. In this implementation, a reset entry will do so.</p>	0b0
[0]	SEO	None	x

Accessibility

Component	Offset	Instance	Range
Debug	0x098	EDECCR	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ERROR

B.2.2.1.10 OSLAR_EL1, OS Lock Access Register

Used to lock or unlock the OS Lock.

Configurations

External debug accesses to OSLAR_EL1 are ignored and return an error when AllowExternalDebugAccess() returns FALSE for the access.

External register OSLAR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.11 OSLAR_EL1, OS Lock Access Register](#) on page 1054 bits [31:0].

Attributes

Width

32

Component

Debug

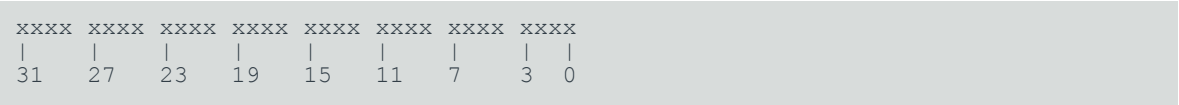
Register offset

0x300

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-116: EXT_OSLAR_EL1 bit assignments

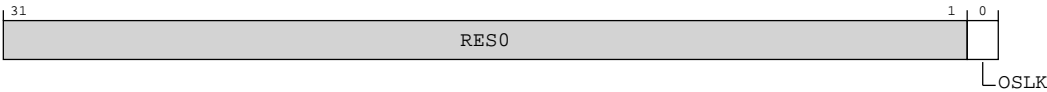


Table B-240: OSLAR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	OSLK	On writes to OSLAR_EL1, bit[0] is copied to the OS Lock. Use ext-EDPRSR.OSLK to check the current status of the lock.	x

Accessibility

Component	Offset	Instance	Range
Debug	0x300	OSLAR_EL1	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalDebugAccess()

WO

Otherwise

ERROR

B.2.2.1.11 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

Configurations

All fields in this register are in the Core power domain.

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR_EL1.

Attributes

Width

32

Component

Debug

Register offset

0x310

Access type

inconsistent

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-117: EXT_EDPRCR bit assignments

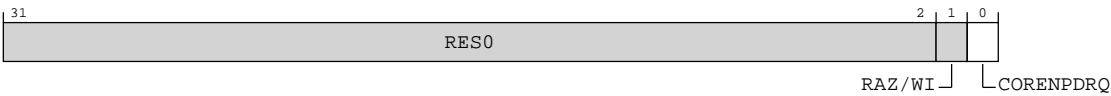


Table B-242: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	RAZ/WI	Reserved	RAZ/ WI

Bits	Name	Description	Reset
[0]	CORENPDRQ	<p>Core no powerdown request. Requests emulation of powerdown.</p> <p>This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1.</p> <p>0b0</p> <p>If the system responds to a powerdown request, it powers down Core power domain.</p> <p>0b1</p> <p>If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.</p> <p>When this bit reads as UNKNOWN, the PE ignores writes to this bit.</p> <p>This field is in the Core power domain, and permitted accesses to this field map to the AArch64-DBGPRCR_EL1.CORENPDRQ field.</p> <p>This bit is not reset to the Cold reset value on exit from an IMPLEMENTATION DEFINED software-visible retention state.</p> <p>Note: Writes to this bit are not prohibited by the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.</p> <p>When OSLockStatus() Access to this field is: UNKNOWN/WI</p> <p>Otherwise Access to this field is: RW</p>	0b0

Accessibility

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

Component	Offset	Instance	Range
Debug	0x310	EDPRCR	None

This interface is accessible as follows:

When IsCorePowered()

RW

Otherwise

ERROR

B.2.2.1.12 EDPRSR, External Debug Processor Status Register

Holds information about the reset and powerdown state of the PE.

Configurations

All fields in this register are in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0x314

Access type

inconsistent

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	x0x0	xxxx	1x11
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-118: EXT_EDPRSR bit assignments

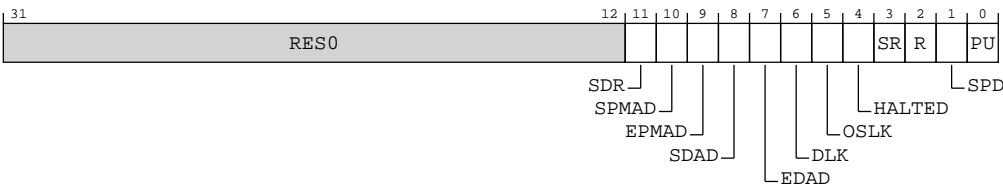


Table B-244: EDPRSR bit descriptions

Bits	Name	Description	Reset
[31:12]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[11]	SDR	<p>Sticky Debug Restart. Set to 1 when the PE exits Debug state.</p> <p>Permitted values are:</p> <p>0b0</p> <p>The PE has not restarted since EDPRSR was last read.</p> <p>0b1</p> <p>The PE has restarted since EDPRSR was last read.</p> <p>Note:</p> <p>If a reset occurs when the PE is in Debug state, the PE exits Debug state. SDR is UNKNOWN on Warm reset, meaning a debugger must also use the SR bit to determine whether the PE has left Debug state.</p> <p>If the Core power domain is powered up, then following a read of EDPRSR:</p> <ul style="list-style-type: none"> This bit clears to 0. <p>This field is in the Core power domain.</p> <p>When ext-EDPRSR.R == '1'</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: RC/WI</p>	x
[10]	SPMAD	<p>Sticky EPMAD error. Set to 1 if an external debug interface access to a Performance Monitors register returns an error because <code>AllowExternalPMUAccess() == FALSE</code>.</p> <p>Permitted values are:</p> <p>0b0</p> <p>No Non-secure external debug interface accesses to the external Performance Monitors registers have failed because <code>AllowExternalPMUAccess() == FALSE</code> for the access since EDPRSR was last read.</p> <p>0b1</p> <p>At least one Non-secure external debug interface access to the external Performance Monitors register has failed and returned an error because <code>AllowExternalPMUAccess() == FALSE</code> for the access since EDPRSR was last read.</p> <p>If the Core power domain is powered up, then following a read of EDPRSR:</p> <ul style="list-style-type: none"> This bit clears to 0. <p>This field is in the Core power domain.</p> <p>When ext-EDPRSR.R == '1'</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: RC/WI</p>	0b0

Bits	Name	Description	Reset
[9]	EPMA	<p>External Performance Monitors Non-secure Access Disable status.</p> <p>0b0</p> <p>External Non-secure Performance Monitors access enabled. <code>AllowExternalPMUAccess()</code> == TRUE for a Non-secure access.</p> <p>0b1</p> <p>External Non-secure Performance Monitors access disabled. <code>AllowExternalPMUAccess()</code> == FALSE for a Non-secure access.</p> <p>This field is in the Core power domain.</p> <p>When ext-EDPRSR.R == '1'</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: RO</p>	x
[8]	SDAD	<p>Sticky EDAD error. Set to 1 if an external debug interface access to a debug register returns an error because <code>AllowExternalDebugAccess()</code> == FALSE.</p> <p>0b0</p> <p>No Non-secure external debug interface accesses to the debug registers have failed because <code>AllowExternalDebugAccess()</code> == FALSE for the access since EDPRSR was last read.</p> <p>0b1</p> <p>At least one Non-secure external debug interface access to the debug registers has failed and returned an error because <code>AllowExternalDebugAccess()</code> == FALSE for the access since EDPRSR was last read.</p> <p>If the Core power domain is powered up, then, following a read of EDPRSR:</p> <ul style="list-style-type: none"> This bit clears to 0. <p>This field is in the Core power domain.</p> <p>When ext-EDPRSR.R == '1'</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: RO</p>	0b0
[7]	EDAD	<p>External Debug Access Disable status.</p> <p>0b0</p> <p>External Non-secure access to breakpoint registers, watchpoint registers, and ext-OSLAR_EL1 enabled. <code>AllowExternalDebugAccess()</code> == TRUE for a Non-secure access.</p> <p>0b1</p> <p>External Non-secure access to breakpoint registers, watchpoint registers, and ext-OSLAR_EL1 disabled. <code>AllowExternalDebugAccess()</code> == FALSE for a Non-secure access.</p> <p>This field is in the Core power domain.</p> <p>When ext-EDPRSR.R == '1'</p> <p>Access to this field is: UNKNOWN/WI</p> <p>Otherwise</p> <p>Access to this field is: RO</p>	x
[6]	DLK	This field is RES0 .	x

Bits	Name	Description	Reset
[5]	OSLK	OS Lock status bit. A read of this bit returns the value of AArch64-OSLSR_EL1.OSLK. This field is in the Core power domain. Access to this field is: RO	x
[4]	HALTED	Halted status bit. 0b0 PE is in Non-debug state. 0b1 PE is in Debug state. This field is in the Core power domain. Access to this field is: RO	x
[3]	SR	Sticky core Reset status bit. Permitted values are: 0b0 The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read. 0b1 The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read. If EDPRSR.PU reads as 1 and EDPRSR.R reads as 0, which means that the Core power domain is in a powerup state and that the non-debug logic of the PE is not in reset state, then following a read of EDPRSR: <ul style="list-style-type: none"> This bit clears to 0. This field is in the Core power domain. Access to this field is: RC/ WI	0b1
[2]	R	PE Reset status bit. Permitted values are: 0b0 The non-debug logic of the PE is not in reset state. 0b1 The non-debug logic of the PE is in reset state. This field is in the Core power domain. Access to this field is: RO	x

Bits	Name	Description	Reset
[1]	SPD	<p>Sticky core Powerdown status bit.</p> <p>For more information, see <i>EDPRSR.{DLK, SPD, PU}</i> and the Core power domain in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>0b0</p> <p>If EDPRSR.PU is 0, it is not known whether the state of the debug registers in the Core power domain is lost.</p> <p>If EDPRSR.PU is 1, the state of the debug registers in the Core power domain has not been lost.</p> <p>0b1</p> <p>The state of the debug registers in the Core power domain has been lost.</p> <p>If the Core power domain is powered up, then, following a read of EDPRSR:</p> <ul style="list-style-type: none"> This bit clears to 0. <p>When <i>EDPRSR.SPD</i> when the Core domain is in either retention or powerdown state in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see <i>EDPRSR.{DLK, SPD, PU}</i> and the Core power domain in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>This field is in the Core power domain.</p> <p>Access to this field is: RO</p>	0b1
[0]	PU	<p>Core powerup status bit.</p> <p>Access to this field is: RAO/WI</p>	0b1

Access

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

If the Core power domain is powered up (EDPRSR.PU == 1), then following a read of EDPRSR:

- EDPRSR.{SDR, SPMAD, SDAD, SPD} are cleared to 0.
- EDPRSR.SR is cleared to 0 if the non-debug logic of the PE is not in reset state (EDPRSR.R == 0).

The clearing of bits is an indirect write to EDPRSR.

Accessibility

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

If the Core power domain is powered up (EDPRSR.PU == 1), then following a read of EDPRSR:

- EDPRSR.{SDR, SPMAD, SDAD, SPD} are cleared to 0.

- EDPRSR.SR is cleared to 0 if the non-debug logic of the PE is not in reset state (EDPRSR.R == 0).

The clearing of bits is an indirect write to EDPRSR.

Component	Offset	Instance	Range
Debug	0x314	EDPRSR	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.13 DBGBVR<n>_EL1, Debug Breakpoint Value Registers, n = 0 - 5

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() && !OSLockStatus() && AllowExternalDebugAccess().
- ERROR otherwise.

External register DBGBVR<n>_EL1 bits [63:0] are architecturally mapped to AArch64 System register A.2.5.6 DBGBVR<n>_EL1, Debug Breakpoint Value Registers, n = 0 - 5 on page 1038 bits [63:0].

Attributes

Width

64

Component

Debug

Register offset

0x400 + (16 * n)

Access type

See bit descriptions

Reset value**When ext-DBGBCR<n>_EL1.BT == '0x0x'**

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

When ext-DBGBCR<n>_EL1.BT == '001x'

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

When ext-DBGBCR<n>_EL1.BT == '011x'

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

When ext-DBGBCR<n>_EL1.BT == '100x'

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

When ext-DBGBCR<n>_EL1.BT == '101x'

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

When ext-DBGBCR<n>_EL1.BT == '110x'

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

When ext-DBGBCR<n>_EL1.BT == '111x'

XXXX XXXX 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-DBGBCR<n>_EL1.BT == '0x0'

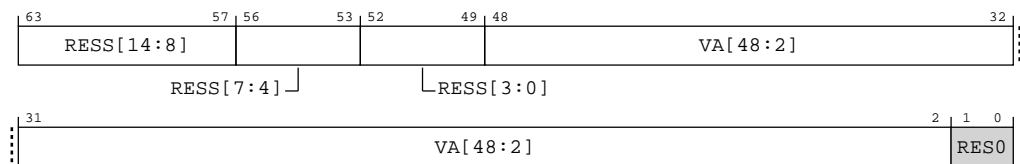
Figure B-119: EXT_DBGBCR<n>_EL1 bit assignments

Table B-246: DBGBVR<n>_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 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.	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: <ul style="list-style-type: none"> This field contains bits[48:2] of the address for comparison. 	47 {x}
[1:0]	RES0	Reserved	RES0

When ext-DBGBCR<n> EL1.BT == '001'

Figure B-120: EXT_DBGBVR<n>_EL1 bit assignments

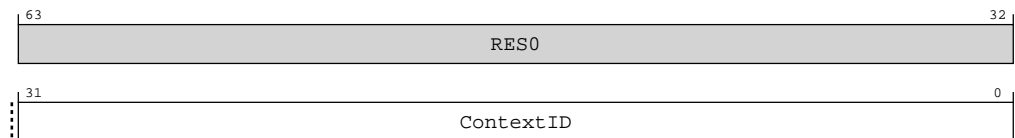


Table B-247: DBGBVR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	<p>Context ID value for comparison.</p> <p>The value is compared against the following:</p> <ul style="list-style-type: none"> AArch64-CONTEXTIDR_EL1 when the PE is executing at AArch64. 	32 { x }

When ext-DBGBCR<n> EL1.BT == '011'

Figure B-121: EXT_DBGBVR<n>_EL1 bit assignments

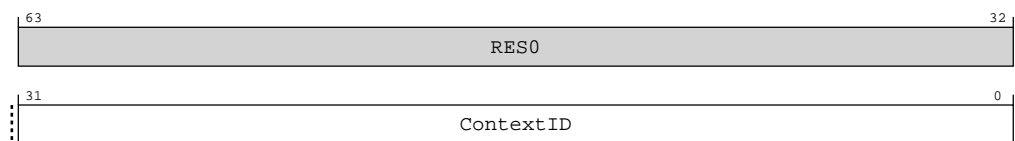
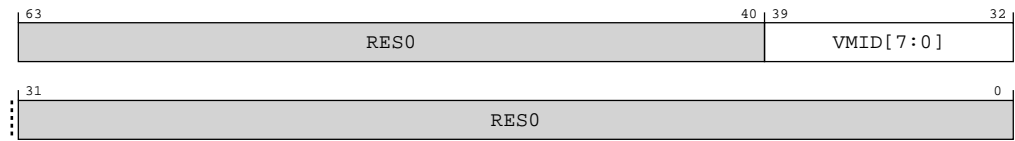


Table B-248: DBGBVR<n>_EL1 bit descriptions

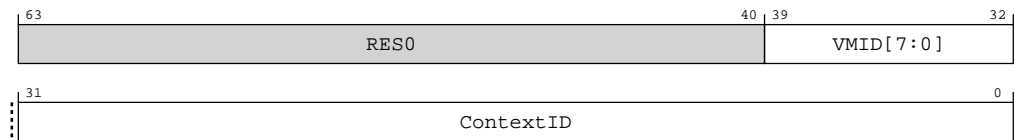
Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32 {x}

When ext-DBGBCR<n> EL1.BT == '100'

Figure B-122: EXT_DBGBVR<n>_EL1 bit assignments**Table B-249: DBGBVR<n>_EL1 bit descriptions**

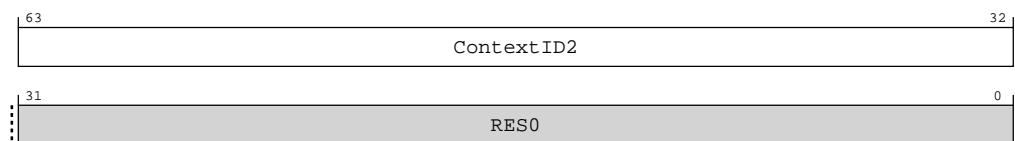
Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39:32]	VMID[7:0]	VMID value for comparison.	8 {x}
[31:0]	RES0	Reserved	RES0

When ext-DBGBCR<n>_EL1.BT == '101'

Figure B-123: EXT_DBGBVR<n>_EL1 bit assignments**Table B-250: DBGBVR<n>_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	RES0
[39:32]	VMID[7:0]	VMID value for comparison.	8 {x}
[31:0]	ContextID	Context ID value for comparison against AArch64-CONTEXTIDR_EL1.	32 {x}

When ext-DBGBCR<n>_EL1.BT == '110'

Figure B-124: EXT_DBGBVR<n>_EL1 bit assignments**Table B-251: DBGBVR<n>_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-DBGBCR<n>_EL1.BT == '111'

Figure B-125: EXT_DBGBVR<n>_EL1 bit assignments

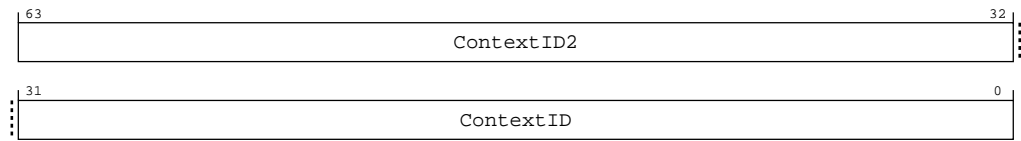


Table B-252: DBGBVR<n>_EL1 bit descriptions

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 }

Accessibility

Component	Offset	Instance	Range
Debug	0x400 + (16 * n)	DBGBVR<n>_EL1	63:0

This interface is accessible as follows:

When IsCorePowered(), !OSLockStatus() and AllowExternalDebugAccess()

RW

Otherwise

ERROR

B.2.2.1.14 DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 5

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() && !OSLockStatus() && AllowExternalDebugAccess().
- ERROR otherwise.

External register DBGBCR<n>_EL1 bits [63:0] are architecturally mapped to AArch64 System register [A.2.5.7 DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 5](#) on page 1043 bits [63:0].

Attributes

Width

64

Component

Debug

Register offset

$0x408 + (16 * n)$

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-126: EXT_DBGBCR<n>_EL1 bit assignments

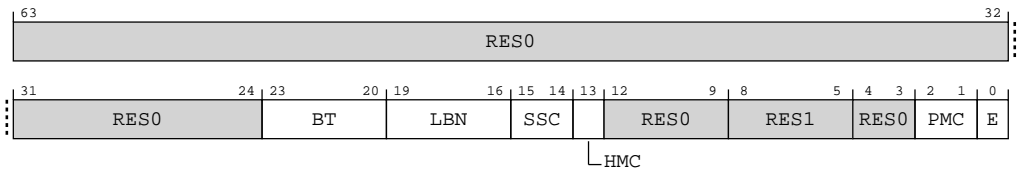


Table B-254: DBGBCR<n>_EL1 bit descriptions

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

Bits	Name	Description	Reset
[23:20]	BT	<p>Breakpoint Type.</p> <p>Specifies breakpoint type.</p> <p>0b0000 Unlinked instruction address match. ext-DBGBVR<n>_EL1 is the address of an instruction.</p> <p>0b0001 Linked instruction address match. As 0b0000, but linked to a breakpoint that has linking enabled.</p> <p>0b0010 Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, the Effective value of AArch64-HCR_EL2.E2H is 1, and either the PE is executing at EL0 with AArch64-HCR_EL2.TGE set to 1 or the PE is executing at EL2, then ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL2 value. Otherwise, ext-DBGBVR<n>_EL1.ContextID must match the AArch64-CONTEXTIDR_EL1 value.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b0011 As 0b0010, with linking enabled.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1000 Unlinked VMID match. ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1001 As 0b1000, with linking enabled.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1010 Unlinked VMID and Context ID match. ext-DBGBVR<n>_EL1.ContextID is a Context ID compared against AArch64-CONTEXTIDR_EL1, and ext-DBGBVR<n>_EL1.VMID is a VMID compared against AArch64-VTTBR_EL2.VMID.</p> <p>This value applies when breakpoint n is context-aware.</p> <p>0b1011 As 0b1010, with linking enabled.</p> <p>This value applies when breakpoint n is context-aware.</p>	xxxx
[19:16]	LBN	<p>Linked Breakpoint Number.</p> <p>For Linked address matching breakpoints, specifies the index of the breakpoint linked to.</p> <p>For all other breakpoint types, this field is ignored and reads of the register return an UNKNOWN value.</p>	xxxx

Bits	Name	Description	Reset
[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 <i>Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values</i> in the Arm® Architecture Reference Manual for A-profile architecture . For more information on the operation of the SSC, HMC, and PMC fields, see <i>Execution conditions for which a breakpoint generates Breakpoint exceptions</i> 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 <i>Execution conditions for which a breakpoint generates Breakpoint exceptions</i> 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 <i>Execution conditions for which a breakpoint generates Breakpoint exceptions</i> 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

Accessibility

Component	Offset	Instance	Range
Debug	0x408 + (16 * n)	DBGBCR<n>_EL1	None

This interface is accessible as follows:

When IsCorePowered(), !OSLockStatus() and AllowExternalDebugAccess()

RW

Otherwise

ERROR

B.2.2.1.15 DBGWVR<n>_EL1, Debug Watchpoint Value Registers, n = 0 - 3

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:

- RES0 when IsCorePowered() && !OSLockStatus() && AllowExternalDebugAccess().
- ERROR otherwise.

External register DBGWVR<n>_EL1 bits [63:0] are architecturally mapped to AArch64 System register A.2.5.8 DBGWVR<n>_EL1, Debug Watchpoint Value Registers, n = 0 - 3 on page 1046 bits [63:0].

Attributes

Width

64

Component

Debug

Register offset

0x800 + (16 * n)

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	0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-127: EXT_DBGWVR<n>_EL1 bit assignments

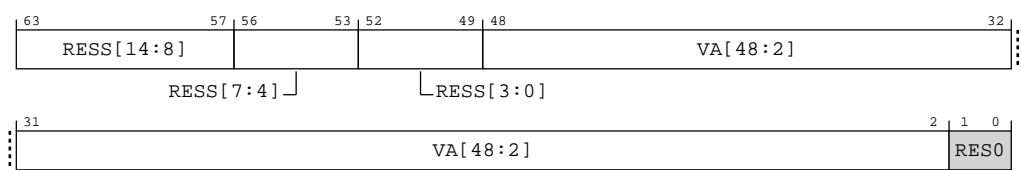


Table B-256: DBGWVR<n>_EL1 bit descriptions

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 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.	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.	4 7 {x}
[1:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
Debug	0x800 + (16 * n)	DBGWVR<n>_EL1	63:0

This interface is accessible as follows:

When IsCorePowered(), !OSLockStatus() and AllowExternalDebugAccess()

RW

Otherwise

ERROR

B.2.2.1.16 DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 3

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:

- RES0 when IsCorePowered() && !OSLockStatus() && AllowExternalDebugAccess().
- ERROR otherwise.

External register DBGWCR<n>_EL1 bits [63:0] are architecturally mapped to AArch64 System register [A.2.5.9 DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 3](#) on page 1048 bits [63:0].

Attributes

Width

64

Component

Debug

Register offset

$0x808 + (16 * n)$

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_DBGWCR<n>_EL1 bit assignments

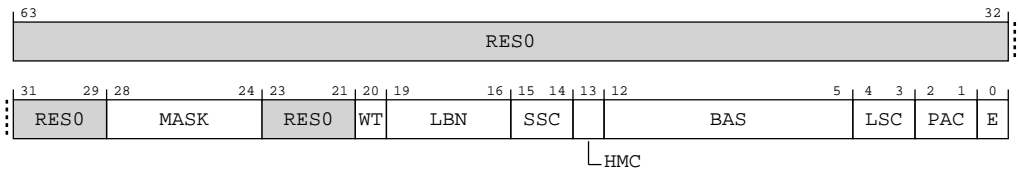


Table B-258: DBGWCR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[63:29]	RES0	Reserved	RES0
[28:24]	MASK	Address Mask. Only objects up to 2GB can be watched using a single mask. 0b00000 No mask. All other values are reserved. Indicates the number of masked address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address). If programmed with a reserved value, the watchpoint behaves as if either: <ul style="list-style-type: none">DBGWCR<n>_EL1.MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCR<n>_EL1.The watchpoint is disabled.	5 {x}
[23:21]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[20]	WT	<p>Watchpoint type. Possible values are:</p> <p>0b0 Unlinked data address match.</p> <p>0b1 Linked data address match.</p>	x
[19:16]	LBN	<p>Linked Breakpoint Number.</p> <p>For Linked data address watchpoints, specifies the index of the breakpoint linked to.</p> <p>For all other watchpoint types, this field is ignored and reads of the register return an UNKNOWN value.</p>	xxxx
[15:14]	SSC	<p>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.</p> <p>For more information on the operation of the SSC, HMC, and PAC fields, see <i>Execution conditions for which a watchpoint generates Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	xx
[13]	HMC	<p>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.</p> <p>For more information on the operation of the SSC, HMC, and PAC fields, see <i>Execution conditions for which a watchpoint generates Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x
[12:5]	BAS	<p>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-259: BAS description on page 1658</p> <p>In cases where ext-DBGWVR<n>_EL1 addresses a double-word: Table B-260: BAS description table 2 on page 1658</p> <p>If ext-DBGWVR<n>_EL1[2] == 1, only BAS[3:0] is used. Arm deprecates setting ext-DBGWVR<n>_EL1[2] == 1.</p> <p>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 <i>Reserved DBGWCR<n>.BAS values</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	8{x}
[4:3]	LSC	<p>Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:</p> <p>0b01 Match instructions that load from a watchpointed address.</p> <p>0b10 Match instructions that store to a watchpointed address.</p> <p>0b11 Match instructions that load from or store to a watchpointed address.</p> <p>All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.</p>	xx
[2:1]	PAC	<p>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.</p> <p>For more information on the operation of the SSC, HMC, and PAC fields, see <i>Execution conditions for which a watchpoint generates Watchpoint exceptions</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	xx

Bits	Name	Description	Reset
[0]	E	Enable watchpoint n. 0b0 Watchpoint n disabled. 0b1 Watchpoint n enabled.	x

Table B-259: 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-260: BAS description table 2

BAS	Description, if AArch64-DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at AArch64-DBGWVR<n>_EL1 + 7

Accessibility

Component	Offset	Instance	Range
Debug	0x808 + (16 * n)	DBGWCR<n>_EL1	None

This interface is accessible as follows:

When IsCorePowered(), !OSLockStatus() and AllowExternalDebugAccess()

RW

Otherwise

ERROR

B.2.2.1.17 MIDR_EL1, Main ID Register

Provides identification information for the PE, including an implementer code for the device and a device ID number.

Configurations

External register MIDR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.1.1 MIDR_EL1, Main ID Register](#) on page 303 bits [31:0].

Attributes

Width

32

Component

Debug

Register offset

0xD00

Access type

See bit descriptions

Reset value

0100 0001 0011 1111 1101 0001 0101 0001

Bit descriptions

Figure B-129: EXT_MIDR_EL1 bit assignments

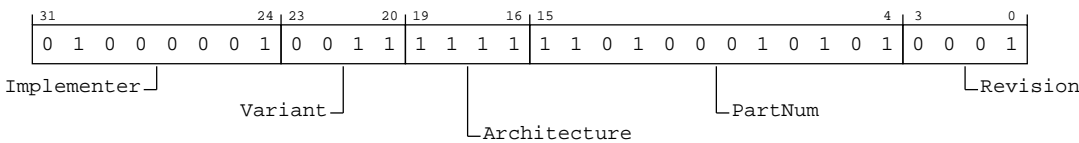


Table B-262: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	Implementer	The Implementer code. This field must hold an implementer code that has been assigned by Arm. 0b01000001 Arm Limited	0x41
[23:20]	Variant	The major product revision variant number. 0b0011 r3.	0b0011
[19:16]	Architecture	Architecture version. Defined values are: 0b1111 Architectural features are individually identified in the ID_* registers. All other values are reserved.	0b1111
[15:4]	PartNum	The primary part number for the device. 0b110100010101 Cortex-R82.	0xD15
[3:0]	Revision	The minor product revision variant number. 0b0001 p1.	0b0001

Accessibility

Component	Offset	Instance	Range
Debug	0xD00	MIDR_EL1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ImplementationDefined

B.2.2.1.18 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, Armv8, for Armv8-A architecture profile.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offsets (2)

0xD20,0xD24

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	0000	xxxx	0001	0000	xxxx	xxxx	xxxx	xxxx	0000	0001	0001	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 B-130: EXT_EDPFR bit assignments

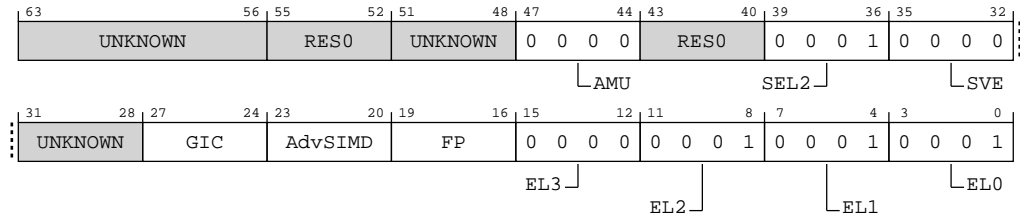


Table B-264: EDPFR bit descriptions

Bits	Name	Description	Reset
[63:56]	UNKNOWN	Reserved	UNKNOWN
[55:52]	RES0	Reserved	RES0
[51:48]	51:48	When Variant(v8Ap4) RES0 Otherwise RES0	xxxx
[47:44]	AMU	Activity Monitors Extension. 0b0000 Activity Monitors Extension is not implemented.	0b0000
[43:40]	RES0	Reserved	RES0
[39:36]	SEL2	Secure EL2. 0b0001 Secure EL2 is implemented.	0b0001
[35:32]	SVE	Scalable Vector Extension. 0b0000 SVE architectural state and programmers' model are not implemented.	0b0000
[31:28]	UNKNOWN	Reserved	UNKNOWN
[27:24]	GIC	When GICCDISABLE == 0 System register GIC interface support. 0001 GIC CPU interface enabled. System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported. Otherwise System register GIC interface support. 0000 GIC CPU interface disabled. System register interface to any external GIC not supported.	xxxx

Bits	Name	Description	Reset
[23:20]	AdvSIMD	When NEON_FPm > 0 Advanced SIMD. 0001 Advanced SIMD is implemented that includes the FEAT_FP16 extension. Otherwise Advanced SIMD. 1111 Advanced SIMD is not implemented.	xxxx
[19:16]	FP	When NEON_FPm > 0 Floating-point. 0001 Floating-point is implemented that includes the FEAT_FP16 extension. Otherwise Floating-point. 1111 Floating-point is not implemented.	xxxx
[15:12]	EL3	EL3 Exception level handling. 0b0000 EL3 is not implemented.	0b0000
[11:8]	EL2	EL2 Exception level handling. 0b0001 EL2 can be executed in AArch64 state only.	0b0001
[7:4]	EL1	EL1 Exception level handling. 0b0001 EL1 can be executed in AArch64 state only.	0b0001
[3:0]	ELO	ELO Exception level handling. 0b0001 ELO can be executed in AArch64 state only.	0b0001

Accessibility

Component	Offset	Instance	Range
Debug	0xD20	EDPFR	31:0

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
Debug	0xD24	EDPFR	63:32

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.19 EDDFR, External Debug Feature Register

Provides top level information about the debug system.

Debuggers must use ext-EDDEVARCH to determine the Debug architecture version. For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers' in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offsets (2)

0xD28,0xD2C

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	0001	xxxx	xxxx	0001	xxxx	0011	xxxx	0101	0101	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-131: EXT_EDDFR bit assignments

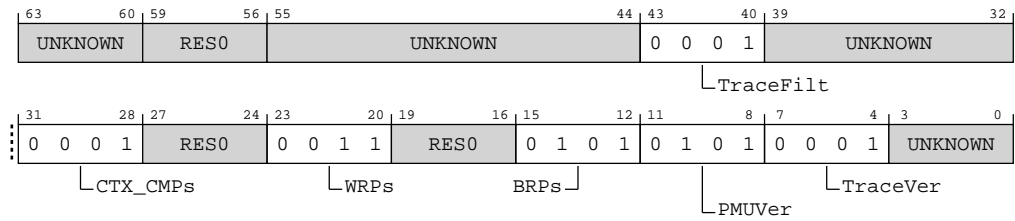


Table B-267: EDDFR bit descriptions

Bits	Name	Description	Reset
[63:60]	UNKNOWN	Reserved	UNKNOWN
[59:56]	RES0	Reserved	RES0
[55:44]	UNKNOWN	Reserved	UNKNOWN
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. 0b0001 Armv8.4 Self-hosted Trace Extension implemented.	0b0001
[39:32]	UNKNOWN	Reserved	UNKNOWN
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. 0b0001 2 context-aware breakpoints implemented.	0b0001
[27:24]	RES0	Reserved	RES0
[23:20]	WRPs	Number of watchpoints, minus 1. 0b0011 4 watchpoints implemented.	0b0011
[19:16]	RES0	Reserved	RES0
[15:12]	BRPs	Number of breakpoints, minus 1. 0b0101 6 breakpoints implemented.	0b0101
[11:8]	PMUVer	Performance Monitors Extension version. 0b0101 PMUv3 for Armv8.4.	0b0101
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE trace unit is implemented. 0b0001 PE trace unit System registers implemented.	0b0001
[3:0]	UNKNOWN	Reserved	UNKNOWN

Accessibility

Component	Offset	Instance	Range
Debug	0xD28	EDDFR	31:0

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

Component	Offset	Instance	Range
Debug	0xD2C	EDDFR	63:32

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.20 EDAA32PFR, External Debug Auxiliary Processor Feature Register

Provides information about implemented PE features.



The register mnemonic, EDAA32PFR, is derived from previous definitions of this register that defined this register only when AArch64 was not supported.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

Debug

Register offset

0xD60

Access type

See bit descriptions

Reset value

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

Figure B-132: EXT_EDAA32PFR bit assignments

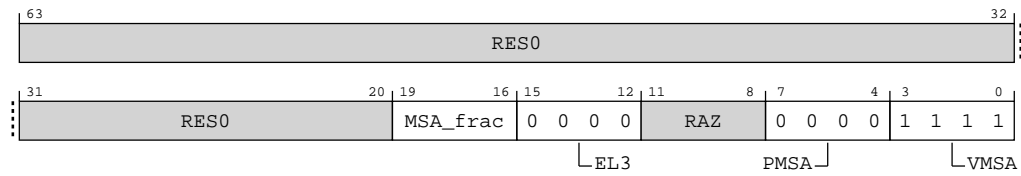


Table B-270: EDAA32PFR bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	RES0
[19:16]	MSA_frac	<p>When VMSAm == 1</p> <p>Memory System Architecture fractional field. This holds information on additional Memory System Architectures supported.</p> <p>0010</p> <p>PMSAv8-64 supported in all translation regimes. In addition to PMSAv8-64, stage 1 EL1&0 transaltion regime also supports VMSAv8-64.</p> <p>Otherwise</p> <p>Memory System Architecture fractional field. This holds information on additional Memory System Architectures supported.</p> <p>0001</p> <p>PMSAv8-64 supported in all translation regimes. VMSAv8-64 not supported.</p>	xxxx
[15:12]	EL3	<p>AArch32 EL3 Exception level handling. Defined values are:</p> <p>0b0000</p> <p>EL3 is not implemented or can be executed in AArch64 state.</p>	0b0000
[11:8]	RAZ	Reserved	RAZ
[7:4]	PMSA	<p>Indicates support for a 32-bit PMSA. Defined values are:</p> <p>0b0000</p> <p>PMSA-32 not supported.</p> <p>All other values are reserved.</p>	0b0000

Bits	Name	Description	Reset
[3:0]	VMSA	Defined values are: 0b1111 Memory system architecture described by EDAA32PFR.MSA_frac. All other values are reserved. In Armv8-R AArch64, the only permitted value is 0b1111.	0b1111

Accessibility

Component	Offset	Instance	Range
Debug	0xD60	EDAA32PFR	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ImplementationDefined

B.2.2.1.21 EDITCTRL, External Debug Integration mode Control register

Enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xF00

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-133: EXT_EDITCTRL bit assignments

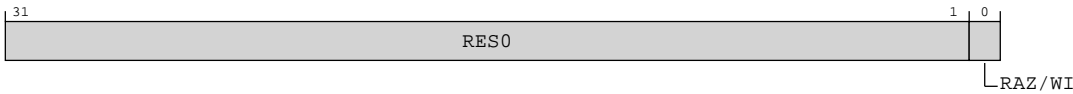


Table B-272: EDITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Instance	Range
Debug	0xF00	EDITCTRL	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ImplementationDefined

B.2.2.1.22 DBGCLAIMSET_EL1, Debug CLAIM Tag Set Register

Used by software to set the CLAIM tag bits to 1.

The architecture does not define any functionality for the CLAIM tag bits.



CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the ext-DBGCLAIMCLR_EL1 register.

Configurations

An implementation must include eight CLAIM tag bits.

External register DBGCLAIMSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.15 DBGCLAIMSET_EL1, Debug CLAIM Tag Set Register](#) on page 1061 bits [31:0].

Attributes

Width

32

Component

Debug

Register offset

0xFA0

Access type

RAOW1S

Reset value

0000 0000 0000 0000 0000 0000 1111 1111

Bit descriptions

Figure B-134: EXT_DBGCLAIMSET_EL1 bit assignments

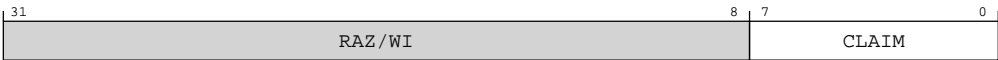


Table B-274: DBGCLAIMSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RAZ/WI	Reserved	RAZ/ WI
[7:0]	CLAIM	Set CLAIM tag bits. This field is RAO . Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1. Writing 0 to one of these bits has no effect.	0xFF

Accessibility

Component	Offset	Instance	Range
Debug	0xFA0	DBGCLAIMSET_EL1	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ERROR

B.2.2.1.23 DBGCLAIMCLR_EL1, Debug CLAIM Tag Clear Register

Used by software to read the values of the CLAIM tag bits, and to clear CLAIM tag bits to 0.

The architecture does not define any functionality for the CLAIM tag bits.



CLAIM tags are typically used for communication between the debugger and target software.

Used in conjunction with the ext-DBGCLAIMSET_EL1 register.

Configurations

An implementation must include eight CLAIM tag bits.

External register DBGCLAIMCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.16 DBGCLAIMCLR_EL1, Debug CLAIM Tag Clear Register](#) on page 1063 bits [31:0].

Attributes

Width

32

Component

Debug

Register offset

0xFA4

Access type

RW1C

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-135: EXT_DBGCLAIMCLR_EL1 bit assignments

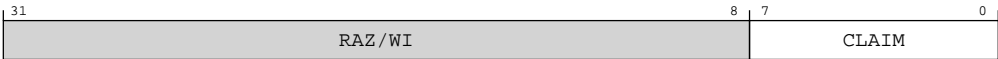


Table B-276: DBGCLAIMCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RAZ/WI	Reserved	RAZ/ WI

Bits	Name	Description	Reset
[7:0]	CLAIM	Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits. Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0. Writing 0 to one of these bits has no effect.	0x00

Accessibility

Component	Offset	Instance	Range
Debug	0xFA4	DBGCLAIMCLR_EL1	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ERROR

B.2.2.1.24 EDDEVAFF0, External Debug Device Affinity register 0

Copy of the low half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFA8

Access type

See bit descriptions

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-136: EXT_EDDEVAFF0 bit assignments

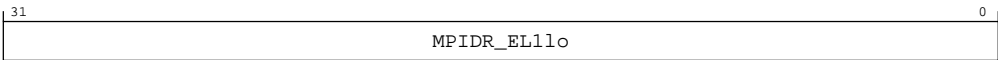


Table B-278: EDDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1lo	AArch64-MPIDR_EL1 low half. Read-only copy of the low half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32 {x}

Accessibility

Component	Offset	Instance	Range
Debug	0xFA8	EDDEVAFF0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.25 EDDEVAFF1, External Debug Device Affinity register 1

Copy of the high half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

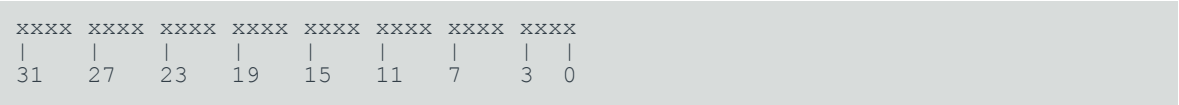
Register offset

0xFAC

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-137: EXT_EDDEVAFF1 bit assignments

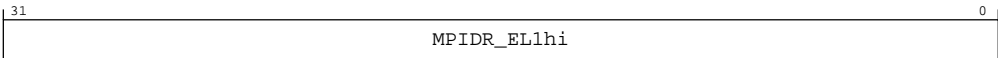


Table B-280: EDDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1hi	AArch64-MPIDR_EL1 high half. Read-only copy of the high half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32 {x}

Accessibility

Component	Offset	Instance	Range
Debug	0xFAC	EDDEVAFF1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.26 EDLAR, External Debug Lock Access Register

Allows or disallows access to the external debug registers through a memory-mapped interface.

The optional Software Lock provides a lock to prevent memory-mapped writes to the debug registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the debug registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFB0

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Otherwise

Figure B-138: EXT_EDLAR bit assignments

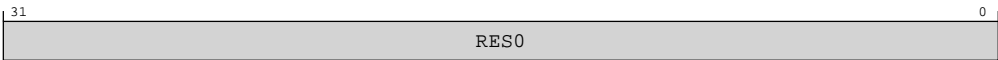


Table B-282: EDLAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
Debug	0xFB0	EDLAR	None

This interface is accessible as follows:

When IsCorePowered()

WO

Otherwise

ERROR

B.2.2.1.27 EDLSR, External Debug Lock Status Register

Indicates the current status of the software lock for external debug registers.

The optional Software Lock provides a lock to prevent memory-mapped writes to the debug registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the debug registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFB4

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx00
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-139: EXT_EDLSR bit assignments

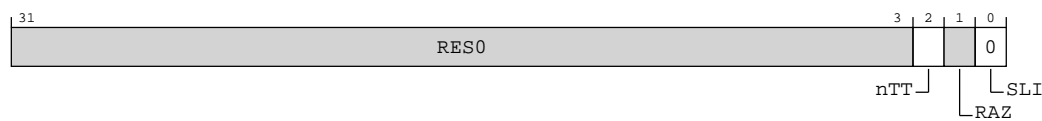


Table B-284: EDLSR bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0
[2]	nTT	Not thirty-two bit access required. RAZ .	x
[1]	RAZ	Reserved	RAZ
[0]	SLI	Indicates whether the Software Lock is implemented. 0b0 Software Lock not implemented or not memory-mapped access.	0b0

Accessibility

Component	Offset	Instance	Range
Debug	0xFB4	EDLSR	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.28 DBGAUTHSTATUS_EL1, Debug Authentication Status Register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for debug.

Configurations

This register is in the Core power domain.

External register DBGAUTHSTATUS_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.5.17 DBGAUTHSTATUS_EL1, Debug Authentication Status Register](#) on page 1065 bits [31:0].

Attributes

Width

32

Component

Debug

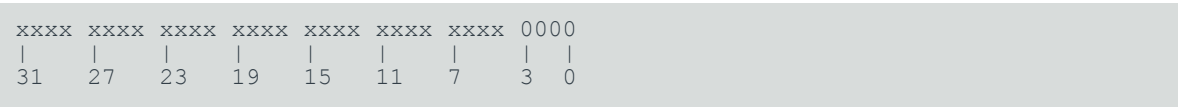
Register offset

0xFB8

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-140: EXT_DBGAUTHSTATUS_EL1 bit assignments



Table B-286: DBGAUTHSTATUS_EL1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure non-invasive debug. This field has the same value as DBGAUTHSTATUS_EL1.SID.	xx
[5:4]	SID	Secure invasive debug. 0b10 Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. 0b11 Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE. All other values are reserved.	xx
[3:2]	NSNID	Non-secure non-invasive debug. 0b00 Non-secure state is not implemented. All other values are reserved.	0b00

Bits	Name	Description	Reset
[1:0]	NSID	Non-secure invasive debug. 0b00 Non-secure state is not implemented. All other values are reserved.	0b00

Accessibility

Component	Offset	Instance	Range
Debug	0xFB8	DBGAUTHSTATUS_EL1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.29 EDDEVARCH, External Debug Device Architecture Register

Identifies the programmers' model architecture of the external debug component.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFBC

Access type

See bit descriptions

Reset value

0100 0111 0111 0000 1001 1010 0000 0101

Bit descriptions

Figure B-141: EXT_EDDEVARCH bit assignments

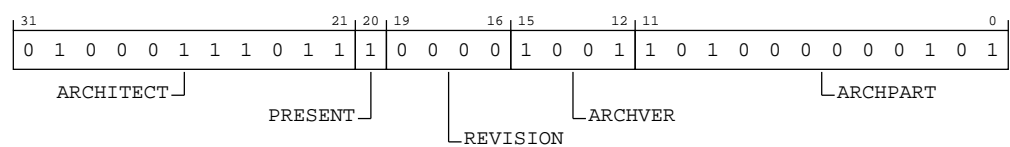


Table B-288: EDDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For debug, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B. 0b01000111011	0b01000111011
[20]	PRESENT	Indicates that the DEVARCH is present. 0b1	0b1
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. For debug, the revision defined by Armv8 is 0x0. All other values are reserved. 0b0000	0b0000
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. Defined values are: 0b1001 Armv8.4 debug architecture, FEAT_Debugv8p4.	0b1001
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. 0b101000000101 Armv8-R debug architecture. EDDEVARCH.ARCHVER and EDDEVARCH.ARCHPART are also defined as a single field, EDDEVARCH.ARCHID, so that EDDEVARCH.ARCHPART is EDDEVARCH.ARCHID[11:0]. Armv8-R debug architecture.	0xA05

Accessibility

Component	Offset	Instance	Range
Debug	0xFBC	EDDEVARCH	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise
ERROR

B.2.2.1.30 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug implementation.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC0

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-142: EXT_EDDEVID2 bit assignments



Table B-290: 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.2.2.1.31 EDDEVID1, External Debug Device ID Register 1

Provides extra information for external debuggers about features of the debug implementation.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFC4

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-143: EXT_EDDEVID1 bit assignments

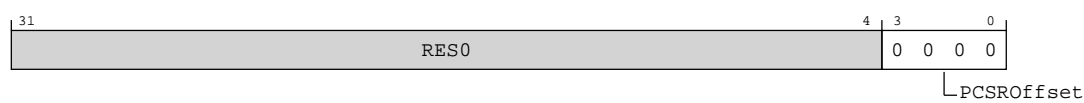


Table B-292: EDDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	PCSROffset	Indicates the offset applied to PC samples returned by reads of ext-EDPCSR. Permitted values of this field in Armv8 are: 0b0000 ext-EDPCSR not implemented.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC4	EDDEVID1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.32 EDDEVID, External Debug Device ID register 0

Provides extra information for external debuggers about features of the debug implementation.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

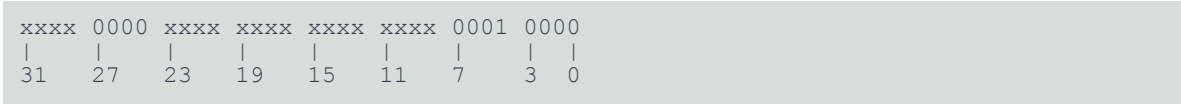
Register offset

0xFC8

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-144: EXT_EDDEVID bit assignments

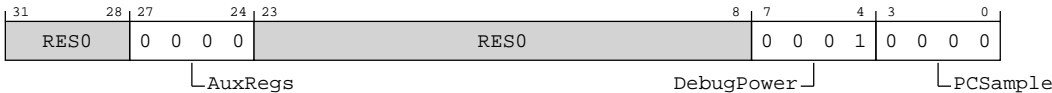


Table B-294: EDDEVID bit descriptions

Bits	Name	Description	Reset
[31:28]	RES0	Reserved	RES0
[27:24]	AuxRegs	Indicates support for Auxiliary registers. Defined values are: 0b0000 None supported.	0b0000
[23:8]	RES0	Reserved	RES0
[7:4]	DebugPower	Indicates support for the FEAT_DoPD feature. Defined values are: 0b0001 FEAT_DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain.	0b0001
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using external debug registers. Defined values are: 0b0000 PC Sample-based Profiling Extension is not implemented in the external debug registers space.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFC8	EDDEVID	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.33 EDDEVTTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PE's debug logic.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

Debug

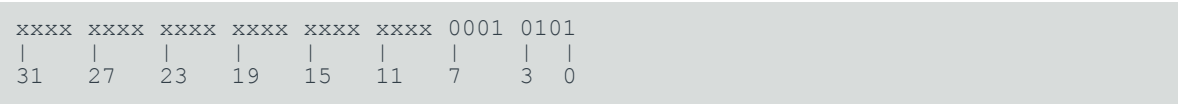
Register offset

0xFCC

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-145: EXT_EDDEVTTYPE bit assignments



Table B-296: EDDEVTTYPE 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.2.2.1.34 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 in the Core power domain.

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-146: EXT_EDPIDR4 bit assignments

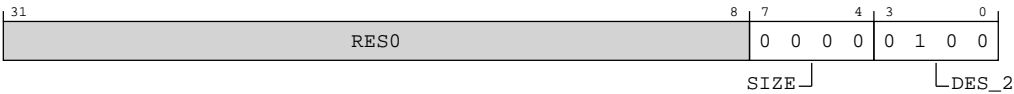


Table B-298: EDPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	Size of the component. RAZ . Log ₂ of the number of 4KB pages from the start of the component to the end of the component ID registers. 0b0000 The component uses a single 4KB block.	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. 0b0100 Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B.	0b0100

Accessibility

Component	Offset	Instance	Range
Debug	0xFD0	EDPIDR4	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.35 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 in the Core power domain.

Attributes

Width

32

Component

Debug

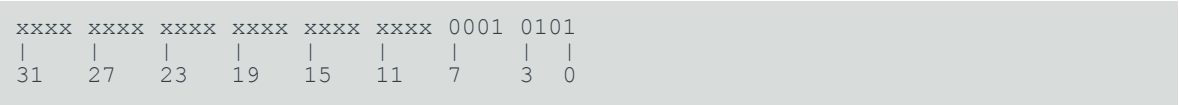
Register offset

0xFE0

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-147: EXT_EDPIDR0 bit assignments

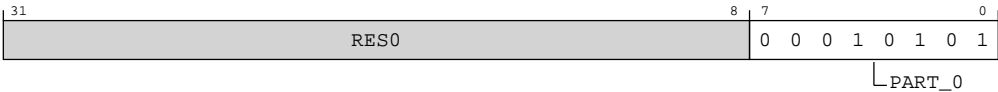


Table B-300: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b00010101 Cortex-R82 external debug. Bits [7:0] of part number 0xD15.	0x15

Accessibility

Component	Offset	Instance	Range
Debug	0xFE0	EDPIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.36 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 in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFE4

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-148: EXT_EDPIDR1 bit assignments

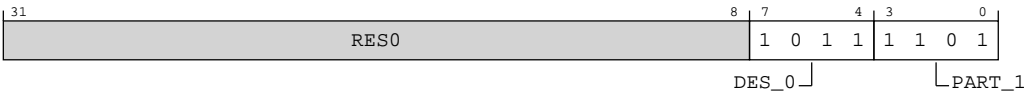


Table B-302: EDPIDR1 bit descriptions

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. Bits [3:0] of JEP106 identification code 0x3B.	0b1011

Bits	Name	Description	Reset
[3:0]	PART_1	Part number bits [11:8]. 0b1101 Cortex-R82 external debug. Bits [11:8] of part number 0xD15.	0b1101

Accessibility

Component	Offset	Instance	Range
Debug	0xFE4	EDPIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.37 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 in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFE8

Access type

See bit descriptions

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0111	1011
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-149: EXT_EDPIDR2 bit assignments

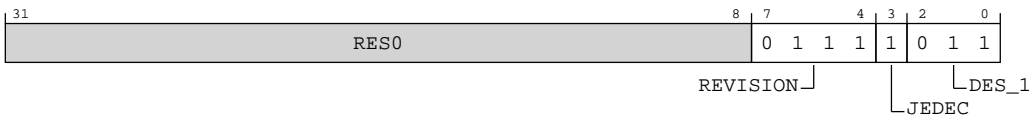


Table B-304: EDPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111
[3]	JEDEC	RAO . Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Instance	Range
Debug	0xFE8	EDPIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.38 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 in the Core power domain.

Attributes

Width

32

Component

Debug

Register offset

0xFEC

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-150: EXT_EDPIDR3 bit assignments

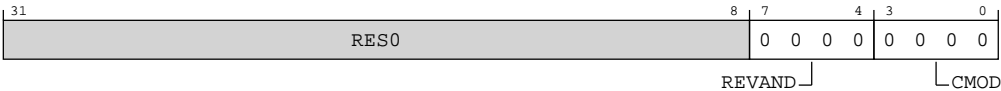


Table B-306: EDPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFEC	EDPIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.39 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 in the Core power domain.

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-151: EXT_EDCIDR0 bit assignments

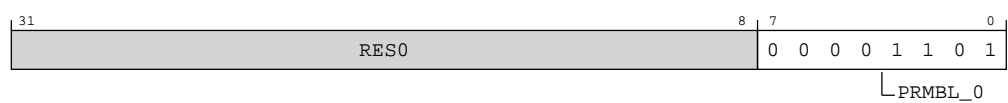


Table B-308: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble. 0b00001101 CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Instance	Range
Debug	0xFF0	EDCIDR0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.40 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 in the Core power domain.

Attributes

Width

32

Component

Debug

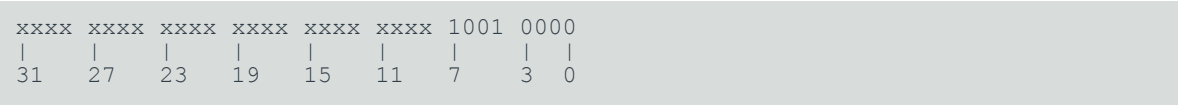
Register offset

0xFF4

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-152: EXT_EDCIDR1 bit assignments

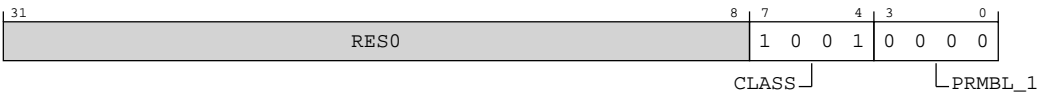


Table B-310: EDCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component class. 0b1001 CoreSight debug component.	0b1001
[3:0]	PRMBL_1	Preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Instance	Range
Debug	0xFF4	EDCIDR1	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.41 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 in the Core power domain.

Attributes

Width

32

Component

Debug

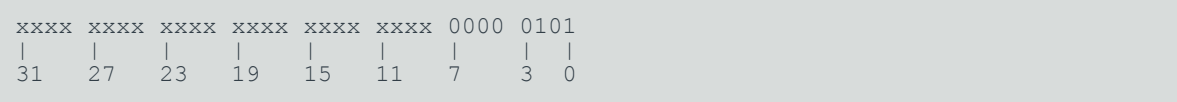
Register offset

0xFF8

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-153: EXT_EDCIDR2 bit assignments

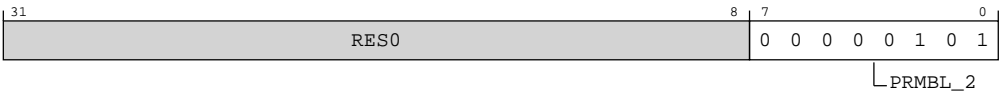


Table B-312: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble. 0b000000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Instance	Range
Debug	0xFF8	EDCIDR2	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.1.42 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 in the Core power domain.

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-154: EXT_EDCIDR3 bit assignments

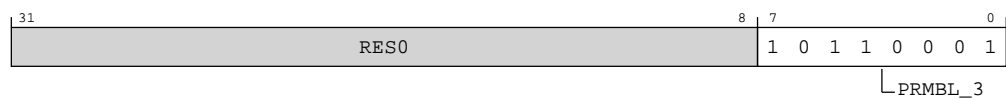


Table B-314: EDCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble. 0b10110001 CoreSight component identification preamble.	0xB1

Accessibility

Component	Offset	Instance	Range
Debug	0xFFC	EDCIDR3	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2 External CLUSTERPMU register description

This section includes the register descriptions for all memory-mapped CLUSTERPMU registers that are accessed for the cluster.

B.2.2.2.1 CLUSTERPMU_PMEVCNTR<n>_EL1, Cluster Performance Monitors Event Count Registers, n = 0 - 5

Holds event counter n, which counts events, where n is 0 to 30.

Configurations

This register is available in all configurations.

Attributes

Width

32


Component
CLUSTERPMU

Register offset
0x000 + (8 * n)

Access type
See bit descriptions

Reset value



 Where the reset reads xxxx, see individual bits.

Note

Bit descriptions

Figure B-155: EXT_CLUSTERPMU_PMEVCNTR<n>_EL1 bit assignments

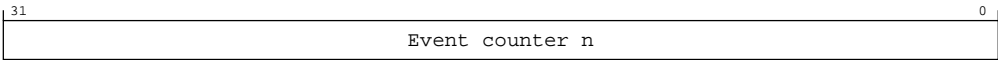


Table B-316: CLUSTERPMU_PMEVCNTR<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[31:0]	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.	32 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0x000 + (8 * n)	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()
RW

Otherwise
ERROR

B.2.2.2.2 CLUSTERPMU_PMCCNTR_EL1, Cluster Performance Monitors Cycle Counter

Holds the value of the Cluster Cycle Counter, CCNT, that counts processor clock cycles.

Configurations

External register CLUSTERPMU_PMCCNTR_EL1 bits [63:0] are architecturally mapped to AArch64 System register [A.2.3.12 IMP_CLUSTERPMCCNTR_EL1, Cluster Performance Monitors Cycle Count Register](#) on page 735 bits [63:0].

Attributes

Width

64

Component

CLUSTERPMU

Register offsets (2)

0x0F8,0x0FC

Access type

See bit descriptions

Reset value

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-156: EXT_CLUSTERPMU_PMCCNTR_EL1 bit assignments

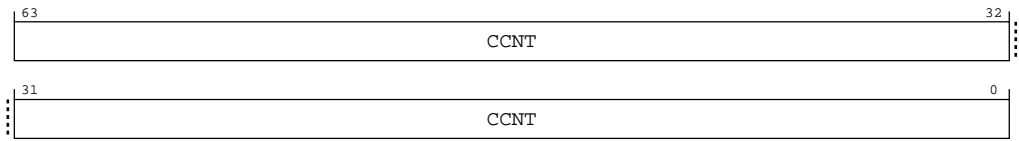


Table B-318: CLUSTERPMU_PMCCNTR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	CCNT	Cycle count. The cycle count increments in every processor clock cycle. Writing 1 to ext-CLUSTERPMU_PMCRR_EL1.C sets this field to 0.	64 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0x0F8	31:0

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()
RW

Otherwise
ERROR

Component	Offset	Range
CLUSTERPMU	0x0FC	63:32

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()
RW

Otherwise
ERROR

B.2.2.2.3 CLUSTERPMU_PMEVTYPEPER<n>_EL1, Cluster Performance Monitors Event Type Registers, n = 0 - 5

Configures event counter n, where n is 0 to 30.

Configurations

If event counter n is not implemented then accesses to this register are RES0.

Attributes

Width
32

Component
CLUSTERPMU

Register offset
 $0x400 + (4 * n)$

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-157: EXT_CLUSTERPMU_PMEVTYPEPER<n>_EL1 bit assignments

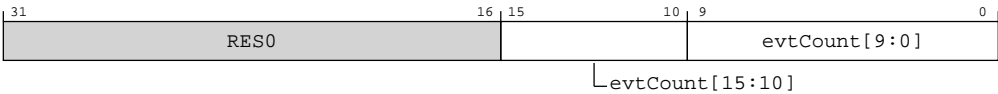


Table B-321: CLUSTERPMU_PMEVTYPEPER<n>_EL1 bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. See evtCount[9:0] for more details.	6 {x}
[9:0]	evtCount[9:0]	<p>Event to count. The event number of the event that is counted by event counter ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>Software must program this field with an event that is supported by the PE being programmed.</p> <p>There are three types of event:</p> <ul style="list-style-type: none">Common architectural and microarchitectural events.Arm recommended common architectural and microarchitectural events.IMPLEMENTATION DEFINED events. <p>The ranges of event numbers allocated to each type of event are shown in .</p> <p>If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type:</p> <ul style="list-style-type: none">For the range 0x000 to 0x03F, no events are counted, and the value returned by a direct or external read of the evtCount field is the value written to the field.For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN. <p>Note: UNPREDICTABLE means the event must not expose privileged information.</p> <p>Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.</p>	10 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0x400 + (4 * n)	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.2.2.2.4 CLUSTERPMU_PMCCFILTR_EL1, Cluster Performance Monitors Cycle Counter Filter Register

Determines the modes in which the Cluster Cycle Counter, ext-CLUSTERPMU_PMCCNTR_EL1, increments. Unlike the per-core PMUs, the Cluster Cycle Counter always increments.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

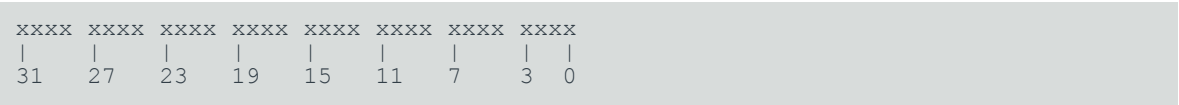
Register offset

0x47C

Access type

See bit descriptions

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-158: EXT_CLUSTERPMU_PMCCFILTR_EL1 bit assignments

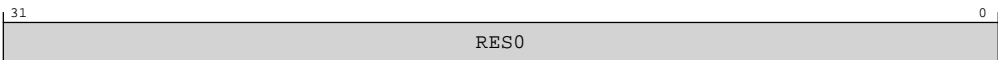


Table B-323: CLUSTERPMU_PMCCFILTR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPMU	0x47C	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.2.2.2.5 CLUSTERPMU_PMCNTENSET_EL1, Cluster Performance Monitors Count Enable Set register

Enables the Cycle Count Register, ext-CLUSTERPMU_PMCCNTR_EL1, and any implemented event counters AArch64-IMP_CLUSTERPMEVCNTR<n>. Reading this register shows which counters are enabled.

Configurations

External register CLUSTERPMU_PMCNTENSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register A.2.3.5 IMP_CLUSTERPMCNTENSET_EL1, Cluster Performance Monitors Count Enable Set register on page 720 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xC00

Access type

RW1S

Reset value



**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-159: EXT_CLUSTERPMU_PMCNTENSET_EL1 bit assignments

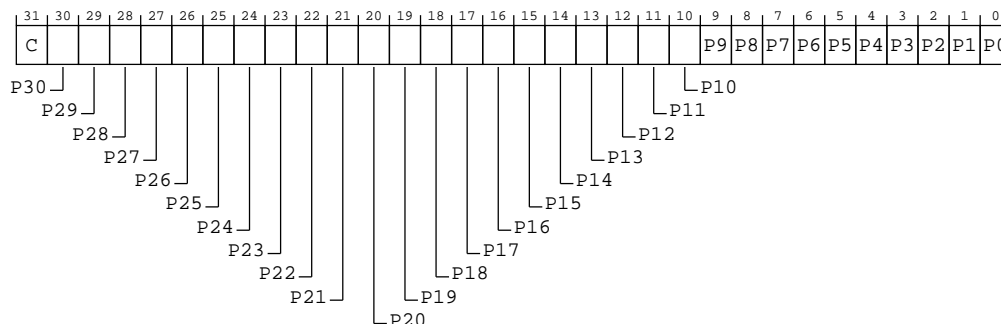


Table B-325: CLUSTERPMU_PMCNTENSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	<p>ext-CLUSTERPMU_PMCNTR_EL1 enable bit. Enables the cycle counter register. Possible values are:</p> <p>0b0</p> <p>When read, means the cycle counter is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter is enabled. When written, enables the cycle counter.</p>	x
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter enable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>If ext-CLUSTERPMU_PMCFGR.N is less than 31, bits [30:ext-CLUSTERPMU_PMCFGR.N] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter is enabled. When written, enables ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p>	31 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0xC00	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise
ERROR

B.2.2.2.6 CLUSTERPMU_PMCNTENCLR_EL1, Cluster Performance Monitors Count Enable Clear register

Disables the Cycle Count Register, ext-CLUSTERPMU_PMCCNTR_EL1, and any implemented event counters AArch64-IMP_CLUSTERPMEVCNTR<n>. Reading this register shows which counters are enabled.

Configurations

External register CLUSTERPMU_PMCNTENCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.6 IMP_CLUSTERPMCNTENCLR_EL1, Cluster Performance Monitors Count Enable Clear register](#) on page 722 bits [31:0].

Attributes

Width
32

Component
CLUSTERPMU

Register offset
0xC20

Access type
RW1C

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-160: EXT_CLUSTERPMU_PMCNTENCLR_EL1 bit assignments

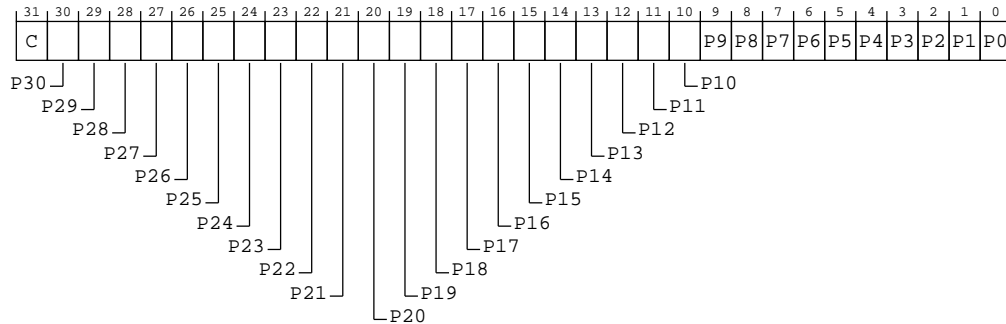


Table B-327: CLUSTERPMU_PMCNTENCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	ext-CLUSTERPMU_PMCNTR_EL1 disable bit. Disables the cycle counter register. Possible values are: 0b0 When read, means the cycle counter is disabled. When written, has no effect. 0b1 When read, means the cycle counter is enabled. When written, disables the cycle counter.	x
[30:0]	P<n>, bit[n], where n = 30 to 0	Event counter disable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1. If ext-CLUSTERPMU_PMCFGR.N is less than 31, bits [30:ext-CLUSTERPMU_PMCFGR.N] are RAZ/WI . 0b0 When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 is disabled. When written, has no effect. 0b1 When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 is enabled. When written, disables ext-CLUSTERPMU_PMEVCNTR<n>_EL1.	31 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0xC20	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.2.2.2.7 CLUSTERPMU_PMINTENSET_EL1, Cluster Performance Monitors Interrupt Enable Set register

Enables the generation of interrupt requests on overflows from the Cycle Count Register, ext-CLUSTERPMU_PMCNTR_EL1, and the event counters ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Reading the register shows which overflow interrupt requests are enabled.

Configurations

External register CLUSTERPMU_PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.10 IMP_CLUSTERPMINTENSET_EL1, Cluster Performance Monitors Interrupt Enable Set register](#) on page 731 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xC40

Access type

RW1S

Reset value

xxxx	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-161: EXT_CLUSTERPMU_PMINTENSET_EL1 bit assignments

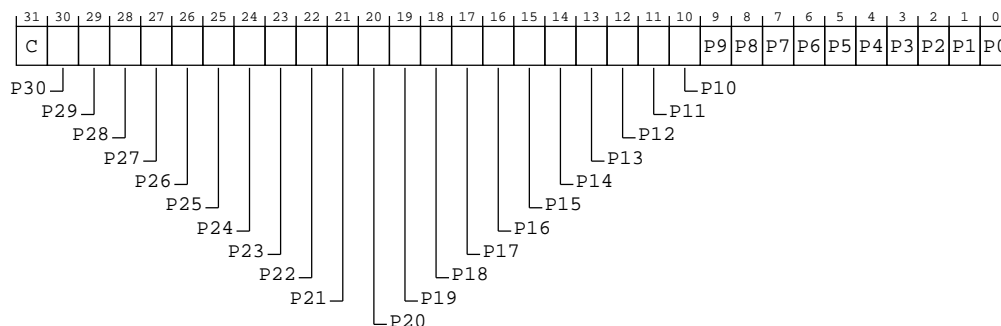


Table B-329: CLUSTERPMU_PMINTENSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	<p>ext-CLUSTERPMU_PMCNTR_EL1 overflow interrupt request enable bit. Possible values are:</p> <p>0b0</p> <p>When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.</p>	x
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow interrupt request enable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>If ext-CLUSTERPMU_PMCNTR.N is less than 31, bits [30:ext-CLUSTERPMU_PMCNTR.N] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is enabled. When written, enables the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 interrupt request.</p>	31 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0xC40	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.2.2.2.8 CLUSTERPMU_PMINTENCLR_EL1, Cluster Performance Monitors Interrupt Enable Clear register

Disables the generation of interrupt requests on overflows from the Cycle Count Register, ext-CLUSTERPMU_PMCNTR_EL1, and the event counters ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Reading the register shows which overflow interrupt requests are enabled.

Configurations

External register CLUSTERPMU_PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.11 IMP_CLUSTERPMINTENCLR_EL1, Cluster Performance Monitors Interrupt Enable Clear register](#) on page 733 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xC60

Access type

RW1C

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-162: EXT_CLUSTERPMU_PMINTENCLR_EL1 bit assignments

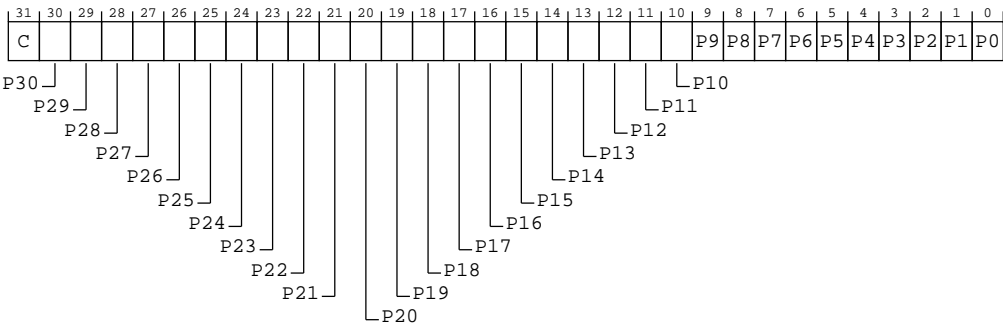


Table B-331: CLUSTERPMU_PMINTENCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	ext-CLUSTERPMU_PMCNTR_EL1 overflow interrupt request disable bit. Possible values are: 0b0 When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect. 0b1 When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.	x

Bits	Name	Description	Reset
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow interrupt request disable bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1.</p> <p>If ext-CLUSTERPMU_PMCFCGR.N is less than 31, bits [30:ext-CLUSTERPMU_PMCFCGR.N] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is disabled. When written, has no effect.</p> <p>0b1</p> <p>When read, means that the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 event counter interrupt request is enabled. When written, disables the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 interrupt request.</p>	31 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0xC60	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.2.2.2.9 CLUSTERPMU_PMOVSLR_EL1, Cluster Performance Monitors Overflow Flag Status Clear register

Contains the state of the overflow bit for the Cycle Count Register, ext-CLUSTERPMU_PMCNTR_EL1, and each of the implemented event counters AArch32-PMEVCNTR<n>. Writing to this register clears these bits.

Configurations

External register CLUSTERPMU_PMOVSLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.8 IMP_CLUSTERPMOVSLR_EL1, Cluster Performance Monitors Overflow Flag Status Clear Register](#) on page 726 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xC80

Access type

RW1C

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-163: EXT_CLUSTERPMU_PMOVSLR_EL1 bit assignments

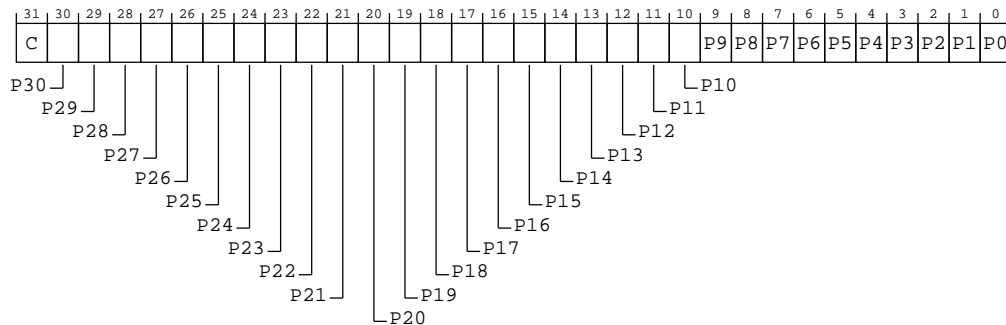


Table B-333: CLUSTERPMU_PMOVSLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	<p>Cycle counter overflow clear bit.</p> <p>0b0</p> <p>When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter has overflowed since this bit was last cleared. When written, clears the cycle counter overflow bit to 0.</p>	x
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow clear bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Bits [30:ext-CLUSTERPMU_PMCFCGR.N] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has overflowed since this bit was last cleared. When written, clears the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 overflow bit to 0.</p>	31 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0xC80	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()
RW

Otherwise
ERROR

B.2.2.2.10 CLUSTERPMU_PMSWINC_EL1, Cluster Performance Monitors Software Increment register

No Software increment events implemented by the Cluster PMU.

Configurations

This register is present only when ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMSWINC_EL1. Otherwise, direct accesses to CLUSTERPMU_PMSWINC_EL1 are UNDEFINED.

Attributes

Width
32

Component
CLUSTERPMU

Register offset
0xCA0

Access type
See bit descriptions

Reset value
0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-164: EXT_CLUSTERPMU_PMSWINC_EL1 bit assignments

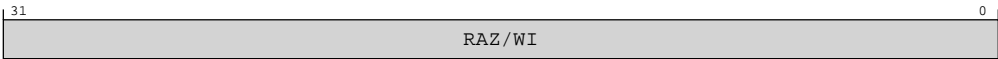


Table B-335: CLUSTERPMU_PMSWINC_EL1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERPMU	0xCA0	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()
WO

Otherwise
ERROR

B.2.2.2.11 CLUSTERPMU_PMOVSET_EL1, Cluster Performance Monitors Overflow Flag Status Set register

Sets the state of the overflow bit for the Cycle Count Register, ext-CLUSTERPMU_PMCNTR_EL1, and each of the implemented event counters AArch32-PMEVCNTR<n>.

Configurations

External register CLUSTERPMU_PMOVSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register A.2.3.7 IMP_CLUSTERPMOVSET_EL1, Cluster Performance Monitors Overflow Flag Status Set register on page 724 bits [31:0].

Attributes

Width
32

Component
CLUSTERPMU

Register offset
0xCC0

Access type
RW1S

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-165: EXT_CLUSTERPMU_PMOVSET_EL1 bit assignments

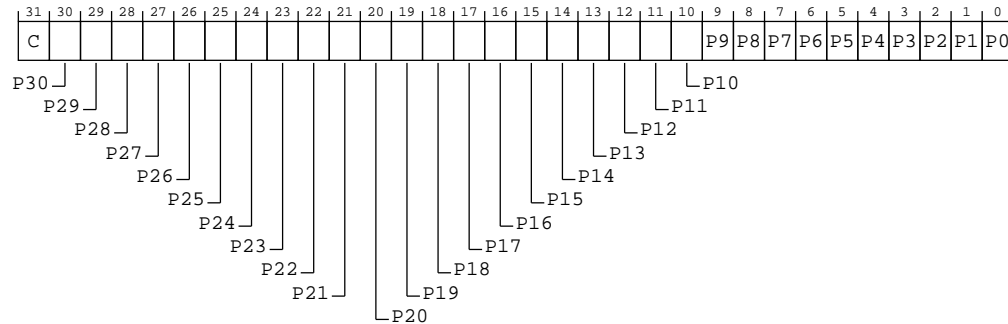


Table B-337: CLUSTERPMU_PMOVSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	<p>Cycle counter overflow set bit.</p> <p>0b0</p> <p>When read, means the cycle counter has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means the cycle counter has overflowed since this bit was last cleared. When written, sets the cycle counter overflow bit to 1.</p>	x
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter overflow set bit for ext-CLUSTERPMU_PMEVCNTR<n>_EL1. Bits [30:ext-CLUSTERPMU_PMCFCGR.N] are RAZ/WI.</p> <p>0b0</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has not overflowed since this bit was last cleared. When written, has no effect.</p> <p>0b1</p> <p>When read, means that ext-CLUSTERPMU_PMEVCNTR<n>_EL1 has overflowed since this bit was last cleared. When written, sets the ext-CLUSTERPMU_PMEVCNTR<n>_EL1 overflow bit to 1.</p>	31 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0xCC0	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.2.2.2.12 CLUSTERPMU_PMCFG, Cluster Performance Monitors Configuration Register

Contains PMU-specific configuration data.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

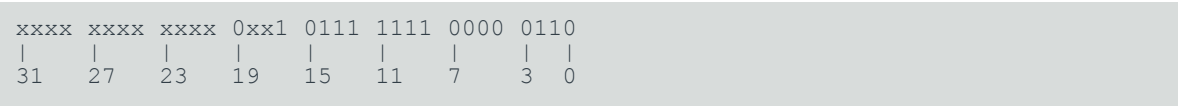
Register offset

0xE00

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-166: EXT_CLUSTERPMU_PMCFG bit assignments

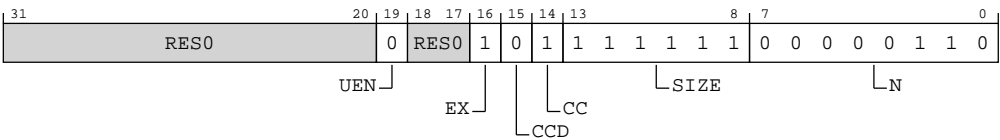


Table B-339: CLUSTERPMU_PMCFG bit descriptions

Bits	Name	Description	Reset
[31:20]	RES0	Reserved	RES0
[19]	UEN	User-mode Enable Register supported. 0b0 User-mode Enable Register not supported.	0b0
[18:17]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[16]	EX	Export supported. 0b1 ext-CLUSTERPMU_PMCR_EL1.X is read/write.	0b1
[15]	CCD	Cycle counter has prescale. 0b0 ext-CLUSTERPMU_PMCR_EL1.D is RES0 .	0b0
[14]	CC	Dedicated cycle counter. 0b1 Dedicated cycle counter ext-CLUSTERPMU_PMCNTR_EL1 is supported.	0b1
[13:8]	SIZE	Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit. This field is used by software to determine the spacing of the counters in the memory-map. 0b111111 The largest counter is 64-bits. Counters are at doubleword-aligned addresses.	0b111111
[7:0]	N	Number of counters implemented in addition to the cycle counter, ext-CLUSTERPMU_PMCNTR_EL1. 0b00000110 ext-CLUSTERPMU_PMCNTR_EL1 plus six event counters implemented.	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	Range
CLUSTERPMU	0xE00	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.2.2.2.13 CLUSTERPMU_PMCR_EL1, Cluster Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configurations

This register is only partially mapped to the internal AArch64-IMP_CLUSTERPMCR System register. An external agent must use other means to discover the information held in AArch64-IMP_CLUSTERPMCR[31:11], such as accessing ext-CLUSTERPMU_PMCFGR and the ID registers.

External register CLUSTERPMU_PMCR_EL1 bits [7:0] are architecturally mapped to AArch64 System register [A.2.3.4 IMP_CLUSTERPMCR_EL1, Cluster Performance Monitors Control Register](#) on page 717 bits [7:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xE04

Access type

inconsistent

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-167: EXT_CLUSTERPMU_PMCR_EL1 bit assignments

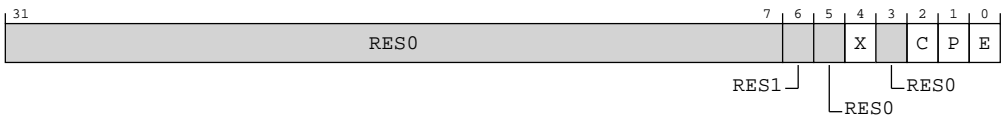


Table B-341: CLUSTERPMU_PMCR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:7]	RES0	Reserved	RES0
[6]	RES1	Reserved	RES1
[5]	RES0	Reserved	RES0
[4]	X	This field enables the exporting of events over an event bus to another device. 0b0 Cluster PMU events are not exported externally.	x
[3]	RES0	Reserved	RES0
[2]	C	Cycle counter reset. This bit is WO. The effects of writing to this bit are: 0b0 No action. 0b1 Reset ext-CLUSTERPMU_PMCCNTR_EL1 to zero. This bit is always RAZ . Note: Resetting ext-CLUSTERPMU_PMCCNTR_EL1 does not change the cycle counter overflow bit.	0b0
[1]	P	Event counter reset. This bit is WO. The effects of writing to this bit are: 0b0 No action. 0b1 Reset all event counters, not including ext-CLUSTERPMU_PMCCNTR_EL1, to zero. This bit is always RAZ . Note: Resetting the event counters does not change the event counter overflow bits.	0b0
[0]	E	Enable. 0b0 All event counters, including ext-CLUSTERPMU_PMCCNTR_EL1, are disabled. 0b1 All event counters can be enabled by ext-CLUSTERPMU_PMCNTENSET_EL1. This bit is RW.	0b0

Accessibility

Component	Offset	Range
CLUSTERPMU	0xE04	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RW

Otherwise

ERROR

B.2.2.2.14 CLUSTERPMU_PMCEID0, Cluster Performance Monitors Common Event Identification register 0

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F

When the value of a bit in the register is 1 the corresponding common event is implemented and counted.

Configurations

External register CLUSTERPMU_PMCEID0 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.15 IMP_CLUSTERPMCEID0_EL1, Cluster Performance Monitors Common Event Identification register 0](#) on page 741 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xE20

Access type

See bit descriptions

Reset value

0110 0110 0000 0010 0000 0000 0000 0000

Bit descriptions

Figure B-168: EXT_CLUSTERPMU_PMCEID0 bit assignments

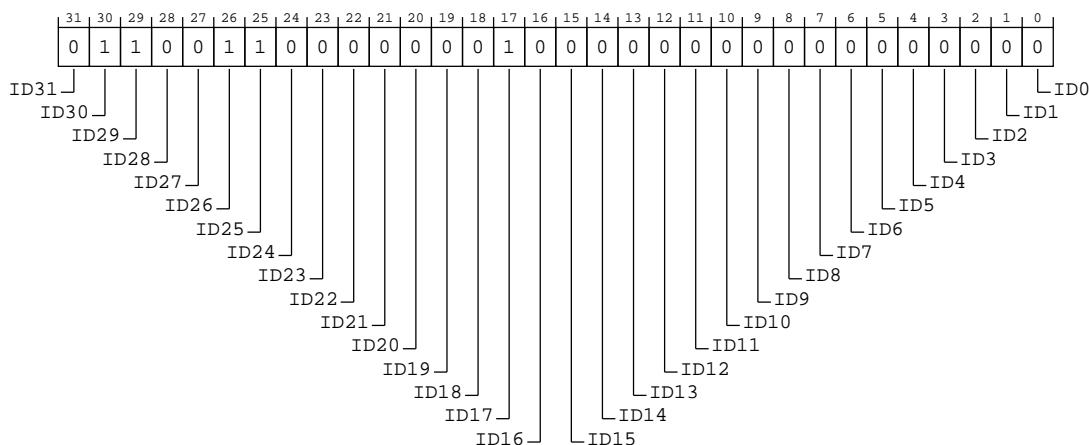


Table B-343: CLUSTERPMU_PMCEID0 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	Common event 0x001F implemented. 0b0 Event 0x001F not implemented.	0b0
[30]	ID30	Common event 0x001E implemented. 0b1 CHAIN event implemented.	0b1
[29]	ID29	Common event 0x001D implemented. 0b1 BUS_CYCLES event implemented.	0b1
[28]	ID28	Common event 0x001C implemented. 0b0 Event 0x001C not implemented.	0b0
[27]	ID27	Common event 0x001B implemented. 0b0 Event 0x001B not implemented.	0b0
[26]	ID26	Common event 0x001A implemented. 0b1 MEMORY_ERROR event implemented.	0b1
[25]	ID25	Common event 0x0019 implemented. 0b1 BUS_ACCESS event implemented.	0b1
[24]	ID24	Common event 0x0018 implemented. 0b0 Event 0x0018 not implemented.	0b0

Bits	Name	Description	Reset
[23]	ID23	Common event 0x0017 implemented. 0b0 Event 0x0017 not implemented.	0b0
[22]	ID22	Common event 0x0016 implemented. 0b0 Event 0x0016 not implemented.	0b0
[21]	ID21	Common event 0x0015 implemented. 0b0 Event 0x0015 not implemented.	0b0
[20]	ID20	Common event 0x0014 implemented. 0b0 Event 0x0014 not implemented.	0b0
[19]	ID19	Common event 0x0013 implemented. 0b0 Event 0x0013 not implemented.	0b0
[18]	ID18	Common event 0x0012 implemented. 0b0 Event 0x0012 not implemented.	0b0
[17]	ID17	Common event 0x0011 implemented. 0b1 CYCLES event implemented.	0b1
[16]	ID16	Common event 0x0010 implemented. 0b0 Event 0x0010 not implemented.	0b0
[15]	ID15	Common event 0x000F implemented. 0b0 Event 0x000F not implemented.	0b0
[14]	ID14	Common event 0x000E implemented. 0b0 Event 0x000E not implemented.	0b0
[13]	ID13	Common event 0x000D implemented. 0b0 Event 0x000D not implemented.	0b0
[12]	ID12	Common event 0x000C implemented. 0b0 Event 0x000C not implemented.	0b0
[11]	ID11	Common event 0x000B implemented. 0b0 Event 0x000B not implemented.	0b0
[10]	ID10	Common event 0x000A implemented. 0b0 Event 0x000A not implemented.	0b0

Bits	Name	Description	Reset
[9]	ID9	Common event 0x0009 implemented. 0b0 Event 0x0009 not implemented.	0b0
[8]	ID8	Common event 0x0008 implemented. 0b0 Event 0x0008 not implemented.	0b0
[7]	ID7	Common event 0x0007 implemented. 0b0 Event 0x0007 not implemented.	0b0
[6]	ID6	Common event 0x0006 implemented. 0b0 Event 0x0006 not implemented.	0b0
[5]	ID5	Common event 0x0005 implemented. 0b0 Event 0x0005 not implemented.	0b0
[4]	ID4	Common event 0x0004 implemented. 0b0 Event 0x0004 not implemented.	0b0
[3]	ID3	Common event 0x0003 implemented. 0b0 Event 0x0003 not implemented.	0b0
[2]	ID2	Common event 0x0002 implemented. 0b0 Event 0x0002 not implemented.	0b0
[1]	ID1	Common event 0x0001 implemented. 0b0 Event 0x0001 not implemented.	0b0
[0]	ID0	Common event 0x0000 implemented. 0b0 Event 0x0000 not implemented.	0b0

Accessibility

Component	Offset	Range
CLUSTERPMU	0xE20	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.2.2.2.15 CLUSTERPMU_PMCEID1, Cluster Performance Monitors Common Event Identification register 1

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted.

Configurations

External register CLUSTERPMU_PMCEID1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.16 IMP_CLUSTERPMCEID1_EL1, Cluster Performance Monitors Common Event Identification register 1](#) on page 748 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xE24

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-169: EXT_CLUSTERPMU_PMCEID1 bit assignments

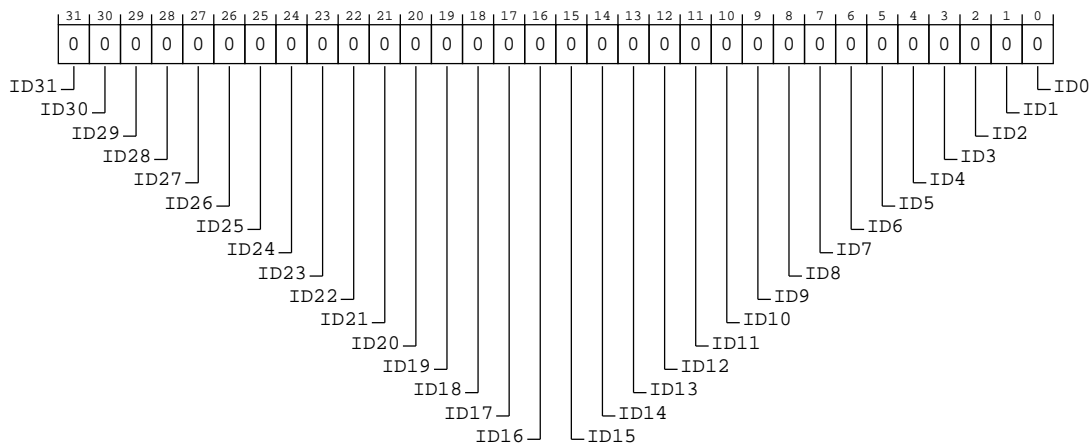


Table B-345: CLUSTERPMU_PMCEID1 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	Common event 0x003F implemented. 0b0 Event 0x003F not implemented.	0b0
[30]	ID30	Common event 0x003E implemented. 0b0 Event 0x003E not implemented.	0b0
[29]	ID29	Common event 0x003D implemented. 0b0 Event 0x003D not implemented.	0b0
[28]	ID28	Common event 0x003C implemented. 0b0 Event 0x003C not implemented.	0b0
[27]	ID27	Common event 0x003B implemented. 0b0 Event 0x003B not implemented.	0b0
[26]	ID26	Common event 0x003A implemented. 0b0 Event 0x003A not implemented.	0b0
[25]	ID25	Common event 0x0039 implemented. 0b0 Event 0x0039 not implemented.	0b0
[24]	ID24	Common event 0x0038 implemented. 0b0 Event 0x0038 not implemented.	0b0
[23]	ID23	Common event 0x0037 implemented. 0b0 Event 0x0037 not implemented.	0b0
[22]	ID22	Common event 0x0036 implemented. 0b0 Event 0x0036 not implemented.	0b0
[21]	ID21	Common event 0x0035 implemented. 0b0 Event 0x0035 not implemented.	0b0
[20]	ID20	Common event 0x0034 implemented. 0b0 Event 0x0034 not implemented.	0b0
[19]	ID19	Common event 0x0033 implemented. 0b0 Event 0x0033 not implemented.	0b0

Bits	Name	Description	Reset
[18]	ID18	Common event 0x0032 implemented. 0b0 Event 0x0032 not implemented.	0b0
[17]	ID17	Common event 0x0031 implemented. 0b0 Event 0x0031 not implemented.	0b0
[16]	ID16	Common event 0x0030 implemented. 0b0 Event 0x0030 not implemented.	0b0
[15]	ID15	Common event 0x002F implemented. 0b0 Event 0x002F not implemented.	0b0
[14]	ID14	Common event 0x002E implemented. 0b0 Event 0x002E not implemented.	0b0
[13]	ID13	Common event 0x002D implemented. 0b0 Event 0x002D not implemented.	0b0
[12]	ID12	Common event 0x002C implemented. 0b0 Event 0x002C not implemented.	0b0
[11]	ID11	Common event 0x002B implemented. 0b0 Event 0x002B not implemented.	0b0
[10]	ID10	Common event 0x002A implemented. 0b0 Event 0x002A not implemented.	0b0
[9]	ID9	Common event 0x0029 implemented. 0b0 Event 0x0029 not implemented.	0b0
[8]	ID8	Common event 0x0028 implemented. 0b0 Event 0x0028 not implemented.	0b0
[7]	ID7	Common event 0x0027 implemented. 0b0 Event 0x0027 not implemented.	0b0
[6]	ID6	Common event 0x0026 implemented. 0b0 Event 0x0026 not implemented.	0b0
[5]	ID5	Common event 0x0025 implemented. 0b0 Event 0x0025 not implemented.	0b0

Bits	Name	Description	Reset
[4]	ID4	Common event 0x0024 implemented. 0b0 Event 0x0024 not implemented.	0b0
[3]	ID3	Common event 0x0023 implemented. 0b0 Event 0x0023 not implemented.	0b0
[2]	ID2	Common event 0x0022 implemented. 0b0 Event 0x0022 not implemented.	0b0
[1]	ID1	Common event 0x0021 implemented. 0b0 Event 0x0021 not implemented.	0b0
[0]	ID0	Common event 0x0020 implemented. 0b0 Event 0x0020 not implemented.	0b0

Accessibility

Component	Offset	Range
CLUSTERPMU	0xE24	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.2.2.2.16 CLUSTERPMU_PMCEID2, Cluster Performance Monitors Common Event Identification register 2

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted.

Configurations

External register CLUSTERPMU_PMCEID2 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.15 IMP_CLUSTERPMCEID0_EL1, Cluster Performance Monitors Common Event Identification register 0](#) on page 741 bits [63:32].

Attributes**Width**

32

Component

CLUSTERPMU

Register offset

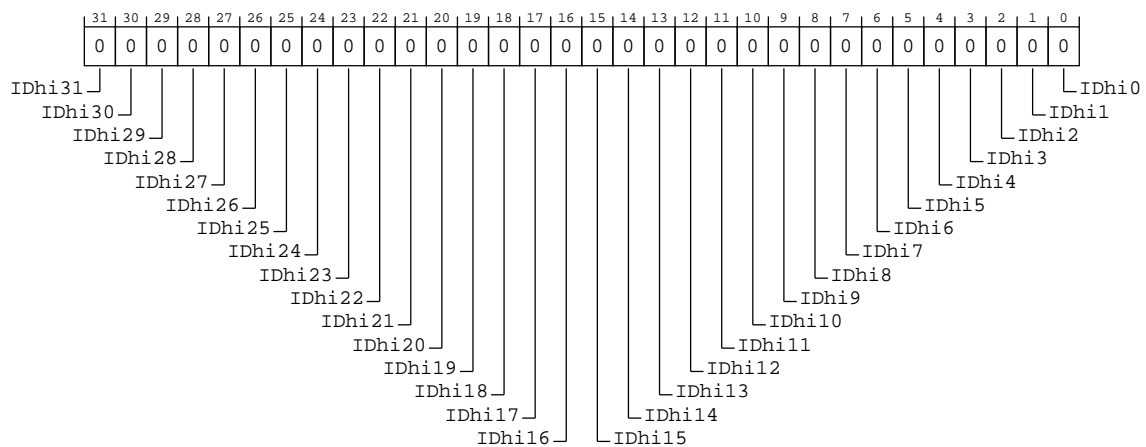
0xE28

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions**Figure B-170: EXT_CLUSTERPMU_PMCEID2 bit assignments****Table B-347: CLUSTERPMU_PMCEID2 bit descriptions**

Bits	Name	Description	Reset
[31]	IDHi31	Common event 0x401F implemented. 0b0 Event 0x401F not implemented.	0b0
[30]	IDHi30	Common event 0x401E implemented. 0b0 Event 0x401E not implemented.	0b0
[29]	IDHi29	Common event 0x401D implemented. 0b0 Event 0x401D not implemented.	0b0

Bits	Name	Description	Reset
[28]	IDHi28	Common event 0x401C implemented. 0b0 Event 0x401C not implemented.	0b0
[27]	IDHi27	Common event 0x401B implemented. 0b0 Event 0x401B not implemented.	0b0
[26]	IDHi26	Common event 0x401A implemented. 0b0 Event 0x401A not implemented.	0b0
[25]	IDHi25	Common event 0x4019 implemented. 0b0 Event 0x4019 not implemented.	0b0
[24]	IDHi24	Common event 0x4018 implemented. 0b0 Event 0x4018 not implemented.	0b0
[23]	IDHi23	Common event 0x4017 implemented. 0b0 Event 0x4017 not implemented.	0b0
[22]	IDHi22	Common event 0x4016 implemented. 0b0 Event 0x4016 not implemented.	0b0
[21]	IDHi21	Common event 0x4015 implemented. 0b0 Event 0x4015 not implemented.	0b0
[20]	IDHi20	Common event 0x4014 implemented. 0b0 Event 0x4014 not implemented.	0b0
[19]	IDHi19	Common event 0x4013 implemented. 0b0 Event 0x4013 not implemented.	0b0
[18]	IDHi18	Common event 0x4012 implemented. 0b0 Event 0x4012 not implemented.	0b0
[17]	IDHi17	Common event 0x4011 implemented. 0b0 Event 0x4011 not implemented.	0b0
[16]	IDHi16	Common event 0x4010 implemented. 0b0 Event 0x4010 not implemented.	0b0
[15]	IDHi15	Common event 0x400F implemented. 0b0 Event 0x400F not implemented.	0b0

Bits	Name	Description	Reset
[14]	IDhi14	Common event 0x400E implemented. 0b0 Event 0x400E not implemented.	0b0
[13]	IDhi13	Common event 0x400D implemented. 0b0 Event 0x400D not implemented.	0b0
[12]	IDhi12	Common event 0x400C implemented. 0b0 Event 0x400C not implemented.	0b0
[11]	IDhi11	Common event 0x400B implemented. 0b0 Event 0x400B not implemented.	0b0
[10]	IDhi10	Common event 0x400A implemented. 0b0 Event 0x400A not implemented.	0b0
[9]	IDhi9	Common event 0x4009 implemented. 0b0 Event 0x4009 not implemented.	0b0
[8]	IDhi8	Common event 0x4008 implemented. 0b0 Event 0x4008 not implemented.	0b0
[7]	IDhi7	Common event 0x4007 implemented. 0b0 Event 0x4007 not implemented.	0b0
[6]	IDhi6	Common event 0x4006 implemented. 0b0 Event 0x4006 not implemented.	0b0
[5]	IDhi5	Common event 0x4005 implemented. 0b0 Event 0x4005 not implemented.	0b0
[4]	IDhi4	Common event 0x4004 implemented. 0b0 Event 0x4004 not implemented.	0b0
[3]	IDhi3	Common event 0x4003 implemented. 0b0 Event 0x4003 not implemented.	0b0
[2]	IDhi2	Common event 0x4002 implemented. 0b0 Event 0x4002 not implemented.	0b0
[1]	IDhi1	Common event 0x4001 implemented. 0b0 Event 0x4001 not implemented.	0b0

Bits	Name	Description	Reset
[0]	IDhi0	Common event 0x4000 implemented. 0b0 Event 0x4000 not implemented.	0b0

Accessibility

Component	Offset	Range
CLUSTERPMU	0xE28	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.2.2.2.17 CLUSTERPMU_PMCEID3, Cluster Performance Monitors Common Event Identification register 3

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted.

Configurations

External register CLUSTERPMU_PMCEID3 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.16 IMP_CLUSTERPMCEID1_EL1, Cluster Performance Monitors Common Event Identification register 1](#) on page 748 bits [63:32].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xE2C

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-171: EXT_CLUSTERPMU_PMCEID3 bit assignments

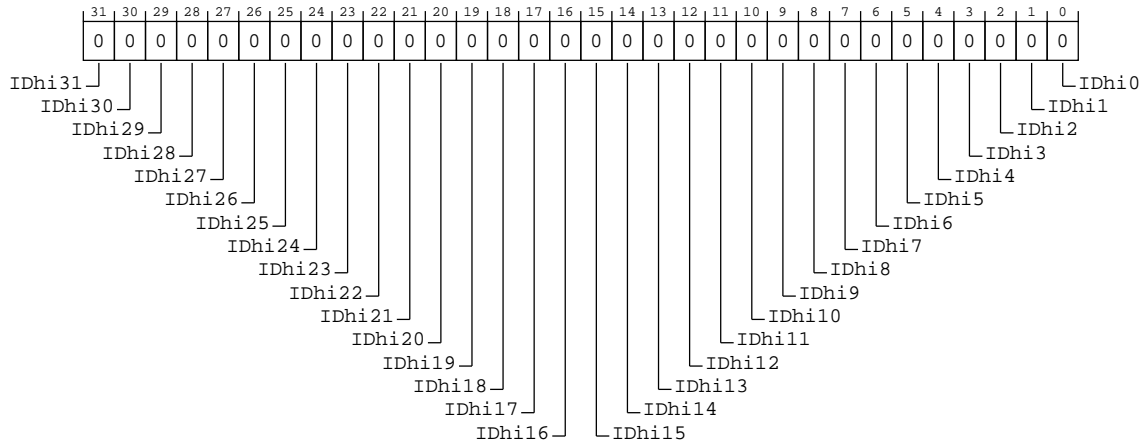


Table B-349: CLUSTERPMU_PMCEID3 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	Common event 0x403F implemented. 0b0 Event 0x403F not implemented.	0b0
[30]	IDhi30	Common event 0x403E implemented. 0b0 Event 0x403E not implemented.	0b0
[29]	IDhi29	Common event 0x403D implemented. 0b0 Event 0x403D not implemented.	0b0
[28]	IDhi28	Common event 0x403C implemented. 0b0 Event 0x403C not implemented.	0b0
[27]	IDhi27	Common event 0x403B implemented. 0b0 Event 0x403B not implemented.	0b0
[26]	IDhi26	Common event 0x403A implemented. 0b0 Event 0x403A not implemented.	0b0
[25]	IDhi25	Common event 0x4039 implemented. 0b0 Event 0x4039 not implemented.	0b0
[24]	IDhi24	Common event 0x4038 implemented. 0b0 Event 0x4038 not implemented.	0b0

Bits	Name	Description	Reset
[23]	IDhi23	Common event 0x4037 implemented. 0b0 Event 0x4037 not implemented.	0b0
[22]	IDhi22	Common event 0x4036 implemented. 0b0 Event 0x4036 not implemented.	0b0
[21]	IDhi21	Common event 0x4035 implemented. 0b0 Event 0x4035 not implemented.	0b0
[20]	IDhi20	Common event 0x4034 implemented. 0b0 Event 0x4034 not implemented.	0b0
[19]	IDhi19	Common event 0x4033 implemented. 0b0 Event 0x4033 not implemented.	0b0
[18]	IDhi18	Common event 0x4032 implemented. 0b0 Event 0x4032 not implemented.	0b0
[17]	IDhi17	Common event 0x4031 implemented. 0b0 Event 0x4031 not implemented.	0b0
[16]	IDhi16	Common event 0x4030 implemented. 0b0 Event 0x4030 not implemented.	0b0
[15]	IDhi15	Common event 0x402F implemented. 0b0 Event 0x402F not implemented.	0b0
[14]	IDhi14	Common event 0x402E implemented. 0b0 Event 0x402E not implemented.	0b0
[13]	IDhi13	Common event 0x402D implemented. 0b0 Event 0x402D not implemented.	0b0
[12]	IDhi12	Common event 0x402C implemented. 0b0 Event 0x402C not implemented.	0b0
[11]	IDhi11	Common event 0x402B implemented. 0b0 Event 0x402B not implemented.	0b0
[10]	IDhi10	Common event 0x402A implemented. 0b0 Event 0x402A not implemented.	0b0

Bits	Name	Description	Reset
[9]	IDhi9	Common event 0x4029 implemented. 0b0 Event 0x4029 not implemented.	0b0
[8]	IDhi8	Common event 0x4028 implemented. 0b0 Event 0x4028 not implemented.	0b0
[7]	IDhi7	Common event 0x4027 implemented. 0b0 Event 0x4027 not implemented.	0b0
[6]	IDhi6	Common event 0x4026 implemented. 0b0 Event 0x4026 not implemented.	0b0
[5]	IDhi5	Common event 0x4025 implemented. 0b0 Event 0x4025 not implemented.	0b0
[4]	IDhi4	Common event 0x4024 implemented. 0b0 Event 0x4024 not implemented.	0b0
[3]	IDhi3	Common event 0x4023 implemented. 0b0 Event 0x4023 not implemented.	0b0
[2]	IDhi2	Common event 0x4022 implemented. 0b0 Event 0x4022 not implemented.	0b0
[1]	IDhi1	Common event 0x4021 implemented. 0b0 Event 0x4021 not implemented.	0b0
[0]	IDhi0	Common event 0x4020 implemented. 0b0 Event 0x4020 not implemented.	0b0

Accessibility

Component	Offset	Range
CLUSTERPMU	0xE2C	None

This interface is accessible as follows:

When IsCorePowered() and AllowExternalPMUAccess()

RO

Otherwise

ERROR

B.2.2.2.18 CLUSTERPMU_PMMIR, Cluster Performance Monitors Machine Identification Register

No STALL_SLOT events for the Cluster PMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xE40

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-172: EXT_CLUSTERPMU_PMMIR bit assignments



Table B-351: CLUSTERPMU_PMMIR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERPMU	0xE40	None

This interface is accessible as follows:

When <code>!IsCorePowered()</code> or <code>!AllowExternalPMUAccess()</code>
ERROR
Otherwise
RO

B.2.2.2.19 CLUSTERPMU_PMITCTRL, Cluster Performance Monitors Integration mode Control register

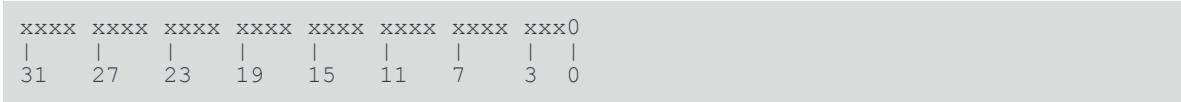
No functional/integration mode switching implemented for the Cluster PMU.

Configurations

This register is present only when `ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMITCTRL`. Otherwise, direct accesses to CLUSTERPMU_PMITCTRL are UNDEFINED.

Attributes

Width
32
Component
CLUSTERPMU
Register offset
0xF00
Access type
See bit descriptions
Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-173: EXT_CLUSTERPMU_PMITCTRL bit assignments

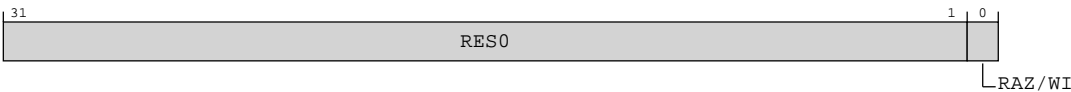


Table B-353: CLUSTERPMU_PMITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERPMU	0xF00	None

This interface is accessible as follows:

When IsCorePowered()

RW

Otherwise

ImplementationDefined

B.2.2.2.20 CLUSTERPMU_PMCLAIMSET_EL1, Cluster Performance Monitors Claim Set register

Used by software to set CLAIM bits to 1.

Configurations

External register CLUSTERPMU_PMCLAIMSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.17 IMP_CLUSTERPMCLAIMSET_EL1, Cluster Performance Monitors Claim Set register](#) on page 755 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFA0

Access type

RW1S

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-174: EXT_CLUSTERPMU_PMCLAIMSET_EL1 bit assignments

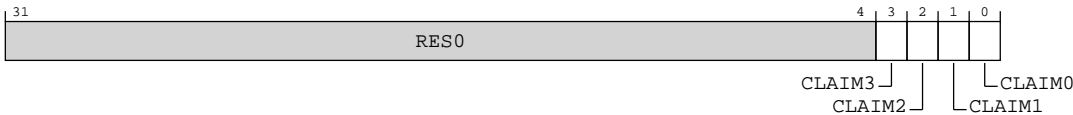


Table B-355: CLUSTERPMU_PMCLAIMSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	CLAIM<x>, bit[x], where x = 3 to 0	CLAIM tag set bit. 0b0 No action. 0b1 Indirectly set claim bit to 1.	xxxx

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFA0	None

This interface is accessible as follows:

When IsCorePowered()

RW

Otherwise

ERROR

B.2.2.2.21 CLUSTERPMU_PMCLAIMCLR_EL1, Cluster Performance Monitors Claim Clear register

Used by software to read the values of the CLAIM bits, and to clear these bits to 0.

Configurations

External register CLUSTERPMU_PMCLAIMCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.18 IMP_CLUSTERPMCLAIMCLR_EL1, Cluster Performance Monitors Claim Clear register](#) on page 757 bits [31:0].

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFA4

Access type

RW1C

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-175: EXT_CLUSTERPMU_PMCLAIMCLR_EL1 bit assignments

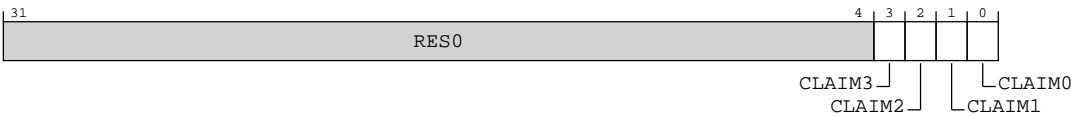


Table B-357: CLUSTERPMU_PMCLAIMCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	CLAIM<x>, bit[x], where x = 3 to 0	CLAIM tag clear bit. 0b0 No action. 0b1 Indirectly clear claim bit to 0.	xxxx

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFA4	None

This interface is accessible as follows:

When IsCorePowered()

RW

Otherwise

ERROR

B.2.2.2.22 CLUSTERPMU_PMDEVAFF0, Cluster Performance Monitors Device Affinity register 0

Allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFA8

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-176: EXT_CLUSTERPMU_PMDEVAFF0 bit assignments

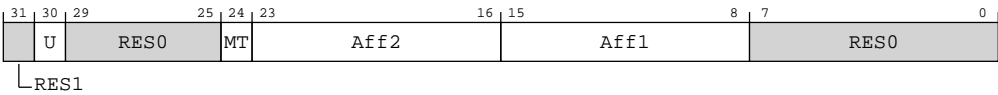


Table B-359: CLUSTERPMU_PMDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31]	RES1	Reserved	RES1
[30]	U	Uniprocessor/Multiprocessor system. 0b0 Processor is part of a multiprocessor system.	x
[29:25]	RES0	Reserved	RES0
[24]	MT	Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. 0b1 Performance of PEs at the lowest affinity level is very interdependent.	x
[23:16]	Aff2	Affinity level 2. Value read from the CFGMPIDRAFF2 configuration pins.	8 {x}
[15:8]	Aff1	Affinity level 1. 0b10000000 Cluster PMU.	8 {x}
[7:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFA8	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.23 CLUSTERPMU_PMDEVAFF1, Cluster Performance Monitors Device Affinity register 1

Allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFAC

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-177: EXT_CLUSTERPMU_PMDEVAFF1 bit assignments



Table B-361: CLUSTERPMU_PMDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	Aff3	Affinity level 3. Value read from the CFGMPIDRAFF3 configuration pins.	8 {x}

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFAC	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.24 CLUSTERPMU_PMLAR, Cluster Performance Monitors Lock Access Register

No software locking implemented for the Cluster PMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFB0

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-178: EXT_CLUSTERPMU_PMLAR bit assignments

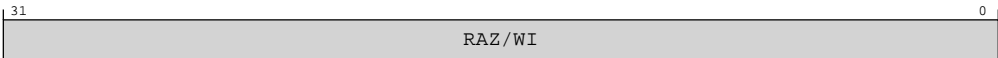


Table B-363: CLUSTERPMU_PMLAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFB0	None

This interface is accessible as follows:

When IsCorePowered()

WO

Otherwise

ERROR

B.2.2.2.25 CLUSTERPMU_PMLSR, Cluster Performance Monitors Lock Status Register

No software locking implemented for the Cluster PMU.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFB4

Access type

See bit descriptions

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-179: EXT_CLUSTERPMU_PMLSR bit assignments

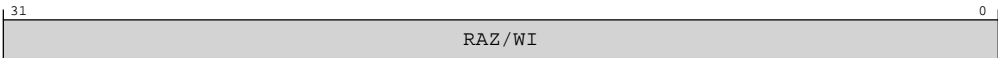


Table B-365: CLUSTERPMU_PMLSR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFB4	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.26 CLUSTERPMU_PMAUTHSTATUS, Cluster Performance Monitors Authentication Status register

Provides information about the state of the authentication interface for Performance Monitors.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

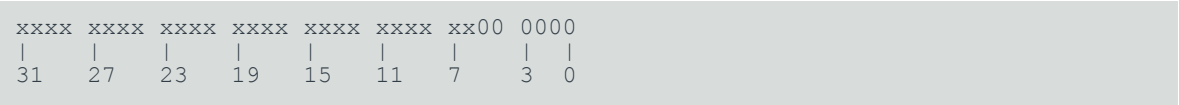
Register offset

0xFB8

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-180: EXT_CLUSTERPMU_PMAUTHSTATUS bit assignments

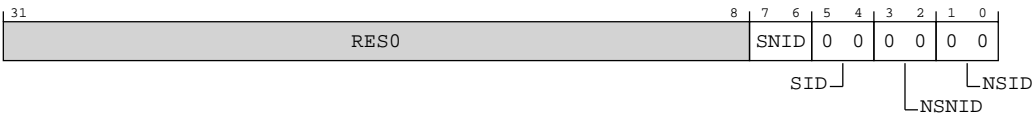


Table B-367: CLUSTERPMU_PMAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure Non-invasive Debug. 0b10 Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE. 0b11 Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.	xx
[5:4]	SID	Secure Invasive Debug. 0b00 Debug level is not supported	0b00
[3:2]	NSNID	Non-secure Non-invasive Debug. 0b00 Debug level is not supported.	0b00

Bits	Name	Description	Reset
[1:0]	NSID	Non-secure Invasive Debug. 0b00 Debug level is not supported.	0b00

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFB8	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.27 CLUSTERPMU_PMDEVARCH, Cluster Performance Monitors Device Architecture register

Identifies the programmers' model architecture of the Performance Monitor component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFBC

Access type

See bit descriptions

Reset value

0100 0111 0111 0000 0010 1010 0001 0110

Bit descriptions

Figure B-181: EXT_CLUSTERPMU_PMDEVARCH bit assignments

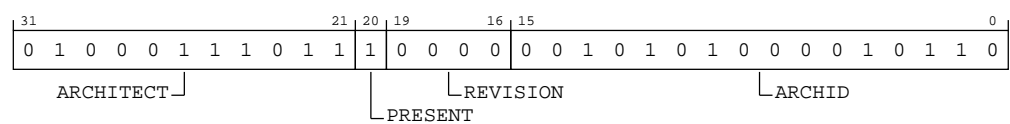


Table B-369: CLUSTERPMU_PMDEVARCH bit descriptions

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. 0b0010101000010110 Processor Performance Monitor (PMU) architecture PMUv3.	0x2A16

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFBC	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.28 CLUSTERPMU_PMDEVID, Cluster Performance Monitors Device ID register

Provides information about features of the Performance Monitors implementation.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERPMU

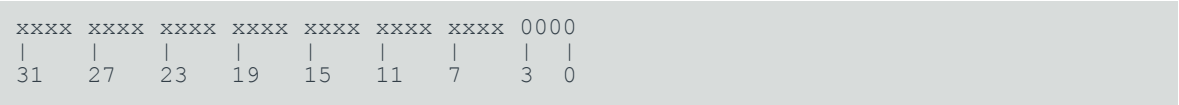
Register offset

0xFC8

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-182: EXT_CLUSTERPMU_PMDEVID bit assignments

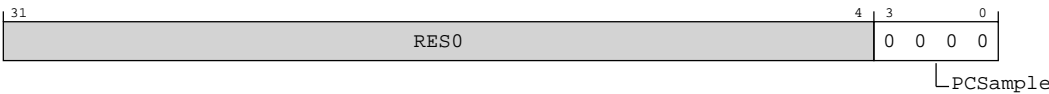


Table B-371: CLUSTERPMU_PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. 0b0000 PC Sample-based Profiling Extension is not implemented for the Cluster PMU.	0b0000

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFC8	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise
ERROR

B.2.2.2.29 CLUSTERPMU_PMDEVTYPE, Cluster Performance Monitors Device Type register

Indicates to a debugger that this component is part of a processor performance monitor interface.

Configurations

This register is present only when `ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMDEVTYPE`. Otherwise, direct accesses to `CLUSTERPMU_PMDEVTYPE` are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFCC

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-183: EXT_CLUSTERPMU_PMDEVTYPE bit assignments



Table B-373: CLUSTERPMU_PMDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	SUB	Subtype. 0b0001 Associated with a processor.	0b0001
[3:0]	MAJOR	Major type. 0b0110 Performance Monitor.	0b0110

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFCC	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.30 CLUSTERPMU_PMPIDR4, Cluster Performance Monitors Peripheral Identification Register 4

Provides information to identify a Performance Monitor component.

For more information see.

Configurations

This register is required for CoreSight compliance.

This register is present only when ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMPIDR4. Otherwise, direct accesses to CLUSTERPMU_PMPIDR4 are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFD0

Access type

See bit descriptions

Reset value

```
xxxx xxxx xxxx xxxx xxxx xxxx 0000 0100
```



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-184: EXT_CLUSTERPMU_PMPIDR4 bit assignments

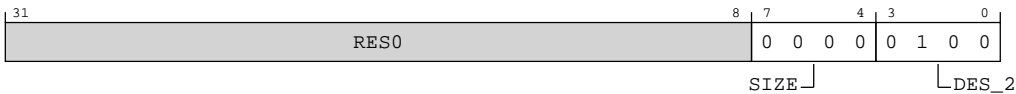


Table B-375: CLUSTERPMU_PMPIDR4 bit descriptions

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
CLUSTERPMU	0xFD0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.31 CLUSTERPMU_PMPIDR0, Cluster Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information see.

Configurations

This register is required for CoreSight compliance.

This register is present only when `ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMPIDR0`. Otherwise, direct accesses to `CLUSTERPMU_PMPIDR0` are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFE0

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-185: EXT_CLUSTERPMU_PMPIDR0 bit assignments



Table B-377: CLUSTERPMU_PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b00010101 Cortex-R82 Cluster PMU. Bits [7:0] of part number 0xD15.	0x15

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFE0	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.32 CLUSTERPMU_PMPIDR1, Cluster Performance Monitors Peripheral Identification Register 1

Provides information to identify a Performance Monitor component.

For more information see.

Configurations

This register is required for CoreSight compliance.

This register is present only when ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMPIDR1. Otherwise, direct accesses to CLUSTERPMU_PMPIDR1 are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

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-186: EXT_CLUSTERPMU_PMPIDR1 bit assignments

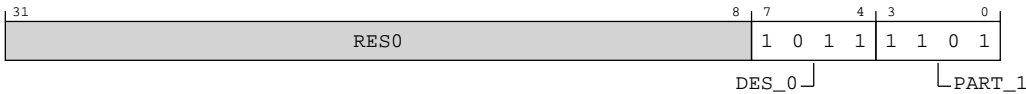


Table B-379: CLUSTERPMU_PMPIDR1 bit descriptions

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 Cortex-R82 Cluster PMU. Bits [11:8] of part number 0xD14.	0b1101

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFE4	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.33 CLUSTERPMU_PMPIDR2, Cluster Performance Monitors Peripheral Identification Register 2

Provides information to identify a Performance Monitor component.

For more information see.

Configurations

This register is required for CoreSight compliance.

This register is present only when ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMPIDR2. Otherwise, direct accesses to CLUSTERPMU_PMPIDR2 are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

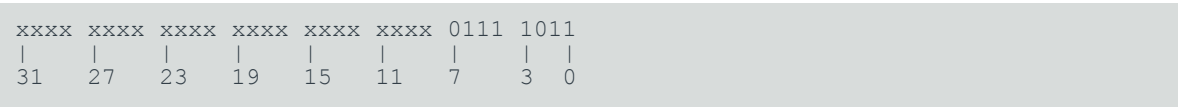
Register offset

0xFE8

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-187: EXT_CLUSTERPMU_PMPIDR2 bit assignments

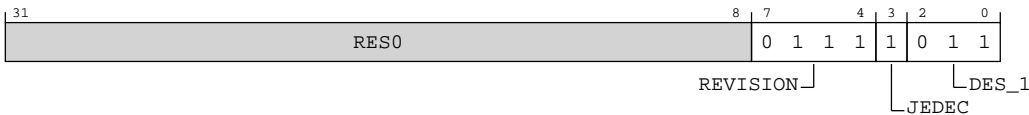


Table B-381: CLUSTERPMU_PMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111
[3]	JEDEC	RAO . Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFE8	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.34 CLUSTERPMU_PMPIDR3, Cluster Performance Monitors Peripheral Identification Register 3

Provides information to identify a Performance Monitor component.

For more information see.

Configurations

This register is required for CoreSight compliance.

This register is present only when `ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMPIDR3`. Otherwise, direct accesses to CLUSTERPMU_PMPIDR3 are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFEC

Access type

See bit descriptions

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-188: EXT_CLUSTERPMU_PMPIDR3 bit assignments



Table B-383: CLUSTERPMU_PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Minor errata fixes. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer Modified. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFEC	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.35 CLUSTERPMU_PMCIDR0, Cluster Performance Monitors Component Identification Register 0

Provides information to identify a Performance Monitor component.

For more information see .

Configurations

This register is present only when ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMCIDR0. Otherwise, direct accesses to CLUSTERPMU_PMCIDR0 are UNDEFINED.

Attributes

Width

32

Component
CLUSTERPMU

Register offset
0xFF0

Access type
See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-189: EXT_CLUSTERPMU_PMCIDR0 bit assignments



Table B-385: CLUSTERPMU_PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble. 0b00001101 CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFF0	None

This interface is accessible as follows:

When IsCorePowered()
RO

Otherwise
ERROR

B.2.2.2.36 CLUSTERPMU_PMCIDR1, Cluster Performance Monitors Component Identification Register 1

Provides information to identify a Performance Monitor component.

For more information see .

Configurations

This register is present only when `ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMCIDR1`. Otherwise, direct accesses to CLUSTERPMU_PMCIDR1 are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

Register offset

0xFF4

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-190: EXT_CLUSTERPMU_PMCIDR1 bit assignments

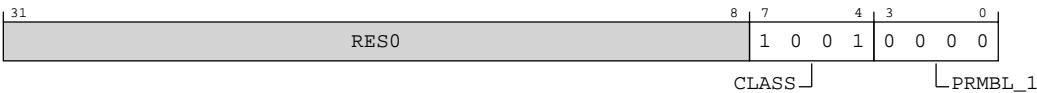


Table B-387: CLUSTERPMU_PMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component class. 0b1001 CoreSight debug component.	0b1001

Bits	Name	Description	Reset
[3:0]	PRMBL_1	Preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFF4	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.37 CLUSTERPMU_PMCIDR2, Cluster Performance Monitors Component Identification Register 2

Provides information to identify a Performance Monitor component.

For more information see .

Configurations

This register is present only when ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMCIDR2. Otherwise, direct accesses to CLUSTERPMU_PMCIDR2 are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-191: EXT_CLUSTERPMU_PMCIDR2 bit assignments

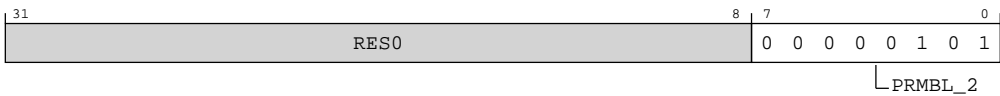


Table B-389: CLUSTERPMU_PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble. 0b00000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFF8	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise

ERROR

B.2.2.2.38 CLUSTERPMU_PMCIDR3, Cluster Performance Monitors Component Identification Register 3

Provides information to identify a Performance Monitor component.

For more information see .

Configurations

This register is present only when ImpDefBool("IMPLEMENTED_CLUSTERPMU_PMCIDR3. Otherwise, direct accesses to CLUSTERPMU_PMCIDR3 are UNDEFINED.

Attributes

Width

32

Component

CLUSTERPMU

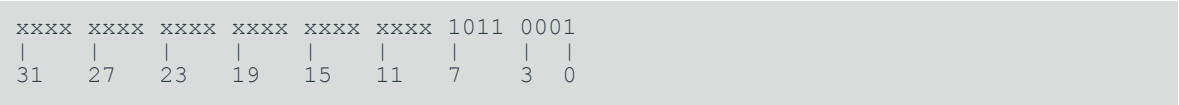
Register offset

0xFFC

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-192: EXT_CLUSTERPMU_PMCIDR3 bit assignments



Table B-391: CLUSTERPMU_PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble. 0b10110001 CoreSight component identification preamble.	0xB1

Accessibility

Component	Offset	Range
CLUSTERPMU	0xFFC	None

This interface is accessible as follows:

When IsCorePowered()

RO

Otherwise
ERROR

B.2.2.3 External DBROM register description

This section includes the register descriptions for all memory-mapped DBROM registers that are accessed for the DebugBlok.

B.2.2.3.1 DBROM_ROMENTRY0, DebugBlock ROM table Entry 0

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x000

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxx0	0000	x111
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-193: EXT_DBROM_ROMENTRY0 bit assignments

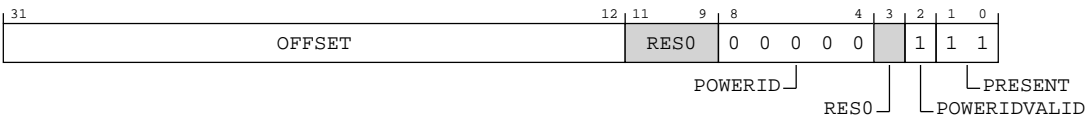


Table B-393: DBROM_ROMENTRY0 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000000001 Cluster ROM table at address 0x0_1000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000001000000000 Cluster ROM table at address 0x20_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00000 PDCLUSTER power domain.</p>	0b00000
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1 The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Accessibility

Component	Offset	Range
DBROM	0x000	None

This interface is accessible as follows:

RO

B.2.2.3.2 DBROM_ROMENTRY1, DebugBlock ROM table Entry 1

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x004

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-194: EXT_DBROM_ROMENTRY1 bit assignments

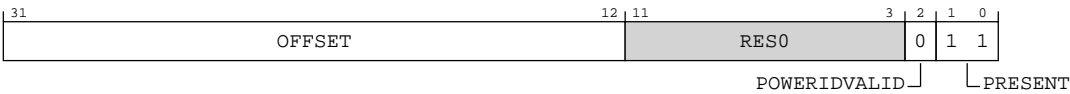


Table B-395: DBROM_ROMENTRY1 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>000000000000000000100 Cluster CTI at address 0x0_4000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000001000110000 Cluster CTI at address 0x23_0000.</p>	20 {x}
[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
DBROM	0x004	None

This interface is accessible as follows:

RO

B.2.2.3.3 DBROM_ROMENTRY2, DebugBlock ROM table Entry 2

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x008

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	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-195: EXT_DBROM_ROMENTRY2 bit assignments

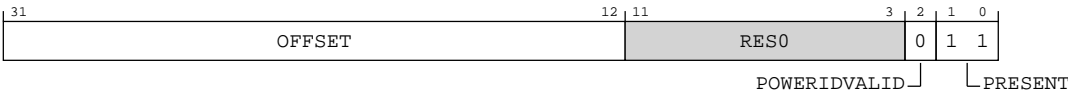


Table B-397: DBROM_ROMENTRY2 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000001010 Core 0 CTI at address 0x0_A000.</p> <p>Otherwise</p> <p>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).</p> <p>0000000000001000000000 Core 0 CTI at address 0x10_0000.</p>	20{x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Accessibility

Component	Offset	Range
DBROM	0x008	None

This interface is accessible as follows:

RO

B.2.2.3.4 DBROM_ROMENTRY3, DebugBlock ROM table Entry 3

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x00C

Access type

RO

Reset value

When NUM_CORES >= 2

xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 2

Figure B-196: EXT_DBROM_ROMENTRY3 bit assignments

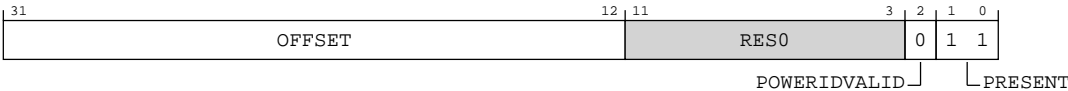


Table B-399: DBROM_ROMENTRY3 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>000000000000000010000 Core 1 CTI at address 0x1_0000.</p> <p>Otherwise</p> <p>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).</p> <p>000000000000100010000 Core 1 CTI at address 0x11_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-197: EXT_DBROM_ROMENTRY3 bit assignments**Table B-400: DBROM_ROMENTRY3 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0x00C	None

This interface is accessible as follows:

RO

B.2.2.3.5 DBROM_ROMENTRY4, DebugBlock ROM table Entry 4

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x010

Access type

RO

Reset value

When NUM_CORES >= 3

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 3

Figure B-198: EXT_DBROM_ROMENTRY4 bit assignments

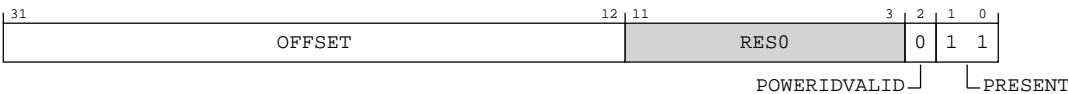
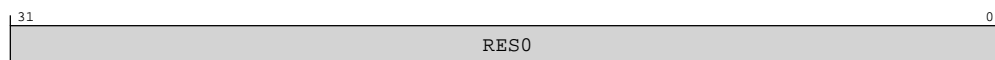


Table B-402: DBROM_ROMENTRY4 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>000000000000000010110 Core 2 CTI at address 0x1_6000.</p> <p>Otherwise</p> <p>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).</p> <p>000000000000100100000 Core 2 CTI at address 0x12_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-199: EXT_DBROM_ROMENTRY4 bit assignments**Table B-403: DBROM_ROMENTRY4 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0x010	None

This interface is accessible as follows:

RO

B.2.2.3.6 DBROM_ROMENTRY5, DebugBlock ROM table Entry 5

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x014

Access type

RO

Reset value

When NUM_CORES >= 4

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 4

Figure B-200: EXT_DBROM_ROMENTRY5 bit assignments

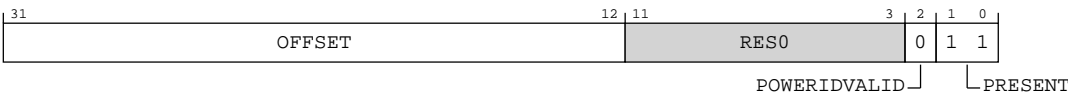
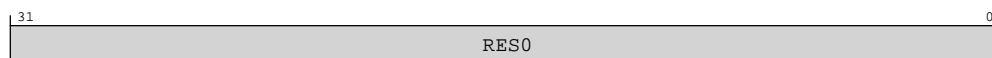


Table B-405: DBROM_ROMENTRY5 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>000000000000000011100 Core 3 CTI at address 0x1_C000.</p> <p>Otherwise</p> <p>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).</p> <p>000000000000100110000 Core 3 CTI at address 0x13_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-201: EXT_DBROM_ROMENTRY5 bit assignments**Table B-406: DBROM_ROMENTRY5 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0x014	None

This interface is accessible as follows:

RO

B.2.2.3.7 DBROM_ROMENTRY6, DebugBlock ROM table Entry 6

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x018

Access type

RO

Reset value

When NUM_CORES >= 5

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 5

Figure B-202: EXT_DBROM_ROMENTRY6 bit assignments

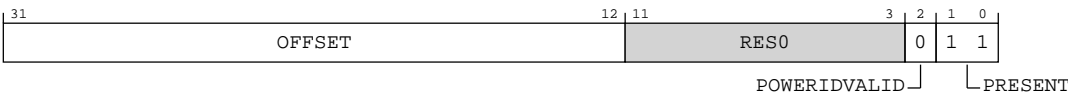


Table B-408: DBROM_ROMENTRY6 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000100010 Core 4 CTI at address 0x2_2000.</p> <p>Otherwise</p> <p>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).</p> <p>000000000000101000000 Core 4 CTI at address 0x14_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-203: EXT_DBROM_ROMENTRY6 bit assignments**Table B-409: DBROM_ROMENTRY6 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0x018	None

This interface is accessible as follows:

RO

B.2.2.3.8 DBROM_ROMENTRY7, DebugBlock ROM table Entry 7

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x01C

Access type

RO

Reset value

When NUM_CORES >= 6

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 6

Figure B-204: EXT_DBROM_ROMENTRY7 bit assignments

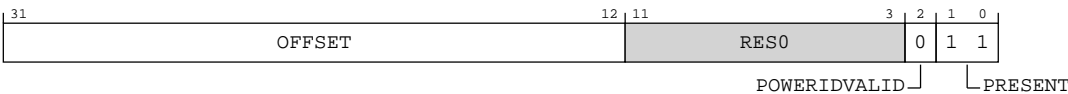


Table B-411: DBROM_ROMENTRY7 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000101000 Core 5 CTI at address 0x2_8000.</p> <p>Otherwise</p> <p>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).</p> <p>000000000000101010000 Core 5 CTI at address 0x15_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-205: EXT_DBROM_ROMENTRY7 bit assignments**Table B-412: DBROM_ROMENTRY7 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0x01C	None

This interface is accessible as follows:

RO

B.2.2.3.9 DBROM_ROMENTRY8, DebugBlock ROM table Entry 8

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x020

Access type

RO

Reset value

When NUM_CORES >= 7

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 7

Figure B-206: EXT_DBROM_ROMENTRY8 bit assignments

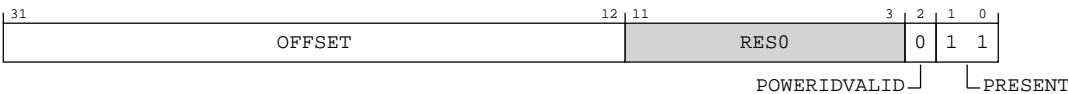
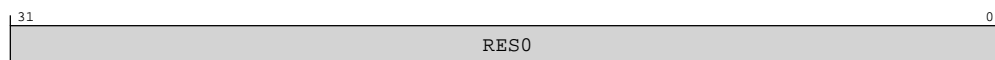


Table B-414: DBROM_ROMENTRY8 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000101110 Core 6 CTI at address 0x2_E000.</p> <p>Otherwise</p> <p>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).</p> <p>000000000000101100000 Core 6 CTI at address 0x16_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-207: EXT_DBROM_ROMENTRY8 bit assignments**Table B-415: DBROM_ROMENTRY8 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0x020	None

This interface is accessible as follows:

RO

B.2.2.3.10 DBROM_ROMENTRY9, DebugBlock ROM table Entry 9

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0x024

Access type

RO

Reset value

When NUM_CORES == 8

xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES == 8

Figure B-208: EXT_DBROM_ROMENTRY9 bit assignments

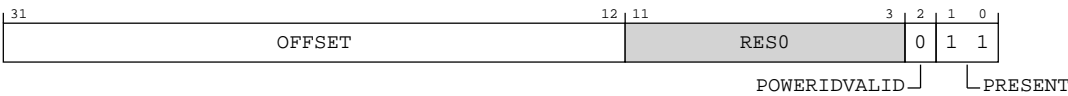
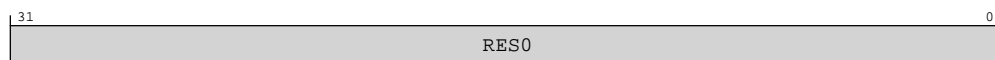


Table B-417: DBROM_ROMENTRY9 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000110100 Core 7 CTI at address 0x3_4000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000000101110000 Core 7 CTI at address 0x17_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-209: EXT_DBROM_ROMENTRY9 bit assignments**Table B-418: DBROM_ROMENTRY9 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0x024	None

This interface is accessible as follows:

RO

B.2.2.3.11 DBROM_DBGPCR0, DebugBlock ROM table Debug Power Control Register 0

Controls power requests for PDCLUSTER.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

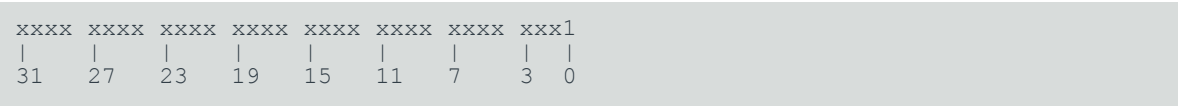
Register offset

0xA00

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-210: EXT_DBROM_DBGPCR0 bit assignments

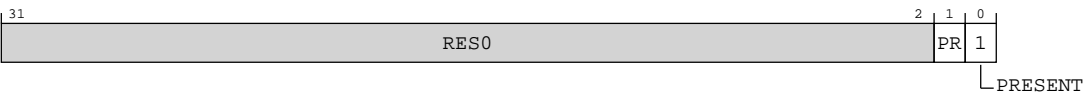


Table B-420: DBROM_DBGPCR0 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCLUSTER. 0b1 Power is requested for PDCLUSTER.	x

Bits	Name	Description	Reset
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCLUSTER is implemented.	0b1

Accessibility

Component	Offset	Range
DBROM	0xA00	None

This interface is accessible as follows:

RW

B.2.2.3.12 DBROM_DBGPSR0, DebugBlock ROM table Debug Power Status Register 0

Indicates the power status for PDCLUSTER.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xA80

Access type

RO

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-211: EXT_DBROM_DBGPSR0 bit assignments

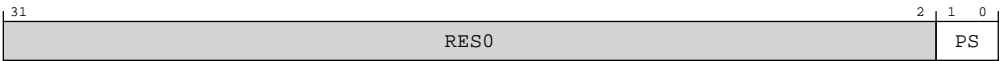


Table B-422: DBROM_DBGPSR0 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	PS	Power Status. 0b00 PDCLUSTER might not be powered. 0b01 PDCLUSTER is powered.	xx

Accessibility

Component	Offset	Range
DBROM	0xA80	None

This interface is accessible as follows:

RO

B.2.2.3.13 DBROM_PRIDR0, DebugBlock ROM table Power Request ID Register 0

Indicates the features of the power request functionality.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xC00

Access type

RO

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xx00 0001



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-212: EXT_DBROM_PRIDR0 bit assignments

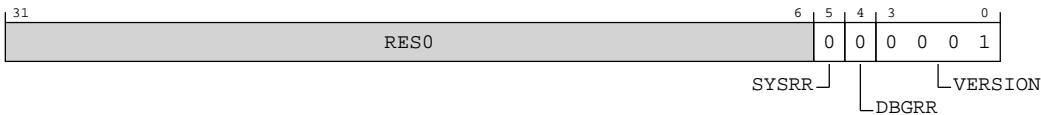


Table B-424: DBROM_PRIDR0 bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	SYSRR	System reset request functionality present. 0b0 The system reset request functionality is not implemented.	0b0
[4]	DBGRR	Debug reset request functionality present. 0b0 The debug reset request functionality is not implemented.	0b0
[3:0]	VERSION	Version of the power request functionality. 0b0001 The power request functionality version 0, and the ext-DBROM_DBGPCRO, ext-DBROM_DBGPSRO, which provide controls for power requests, are implemented.	0b0001

Accessibility

Component	Offset	Range
DBROM	0xC00	None

This interface is accessible as follows:

RO

B.2.2.3.14 DBROM_ITCTRL, DebugBlock ROM table Integration Mode Control Register

No functional/integration mode switching implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
DBROM

Register offset
0xF00

Access type
RW

Reset value

xxxx

xxxx

xxxx

xxxx

xxxx

xxxx

xxxx

xxx0

31

27

23

19

15

11

7

3

0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-213: EXT_DBROM_ITCTRL bit assignments

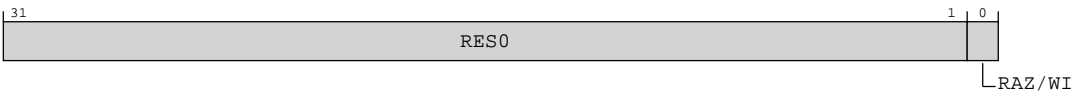


Table B-426: DBROM_ITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
DBROM	0xF00	None

This interface is accessible as follows:

RW

B.2.2.3.15 DBROM_CLAIMSET, DebugBlock ROM table Claim Tag Set Register

No claim tags implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFA0

Access type

RW

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-214: EXT_DBROM_CLAIMSET bit assignments

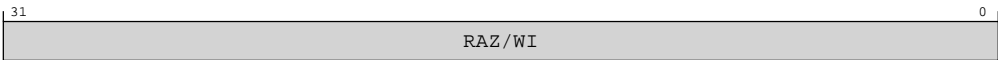


Table B-428: DBROM_CLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
DBROM	0xFA0	None

This interface is accessible as follows:

RW

B.2.2.3.16 DBROM_CLAIMCLR, DebugBlock ROM table Claim Tag Clear Register

No claim tags implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFA4

Access type

RW

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-215: EXT_DBROM_CLAIMCLR bit assignments

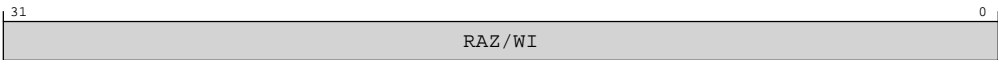


Table B-430: DBROM_CLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
DBROM	0xFA4	None

This interface is accessible as follows:

RW

B.2.2.3.17 DBROM_DEVAFF0, DebugBlock ROM table Device Affinity Register 0

Enables a debugger to determine whether two components have an affinity with each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFA8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-216: EXT_DBROM_DEVAFF0 bit assignments

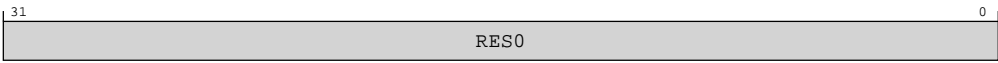


Table B-432: DBROM_DEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0xFA8	None

This interface is accessible as follows:

RO

B.2.2.3.18 DBROM_DEVAFF1, DebugBlock ROM table Device Affinity Register 1

Enables a debugger to determine whether two components have an affinity with each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFAC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-217: EXT_DBROM_DEVAFF1 bit assignments

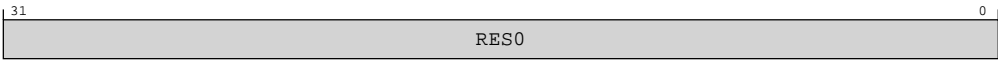


Table B-434: DBROM_DEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0xFAC	None

This interface is accessible as follows:

RO

B.2.2.3.19 DBROM_LAR, DebugBlock ROM table Software Lock Access Register

No software locking implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFB0

Access type

RESERVEDW

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-218: EXT_DBROM_LAR bit assignments



Table B-436: DBROM_LAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
DBROM	0xFB0	None

This interface is accessible as follows:

WO

B.2.2.3.20 DBROM_LSR, DebugBlock ROM table Software Lock Status Register

No software locking implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFB4

Access type

RO

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-219: EXT_DBROM_LSR bit assignments

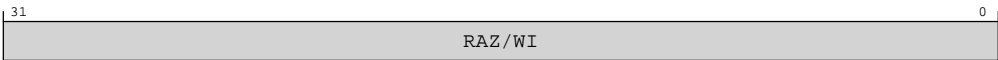


Table B-438: DBROM_LSR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
DBROM	0xFB4	None

This interface is accessible as follows:

RO

B.2.2.3.21 DBROM_AUTHSTATUS, DebugBlock 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

DBROM

Register offset

0xFB8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-220: EXT_DBROM_AUTHSTATUS bit assignments

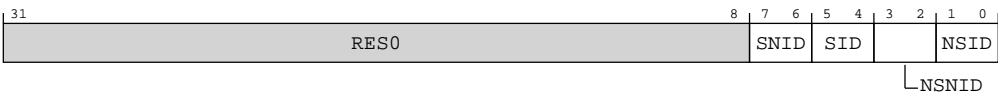


Table B-440: DBROM_AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure Non-invasive Debug. ExternalSecureNoninvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field.	xx

Bits	Name	Description	Reset
[5:4]	SID	Secure Invasive Debug. 0b10 Secure invasive debug disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. 0b11 Secure invasive debug enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.	xx
[3:2]	NSNID	Non-secure Non-invasive Debug. ExternalNoninvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field.	xx
[1:0]	NSID	Non-secure Invasive Debug. ExternalInvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field.	xx

Accessibility

Component	Offset	Range
DBROM	0xFB8	None

This interface is accessible as follows:

RO

B.2.2.3.22 DBROM_DEVARCH, DebugBlock 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

DBROM

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 0000 0000 1010 1111 0111

Bit descriptions

Figure B-221: EXT_DBROM_DEVARCH bit assignments

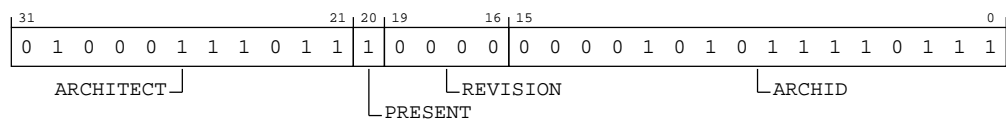


Table B-442: DBROM_DEVARCH bit descriptions

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-DBROM_DEVTYPE and ext-DBROM_DEVID to determine further information about the ROM Table.	0x0AF7

Accessibility

Component	Offset	Range
DBROM	0xFBC	None

This interface is accessible as follows:

RO

B.2.2.3.23 DBROM_DEVID2, DebugBlock ROM table Device Configuration Register 2

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFC0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-222: EXT_DBROM_DEVID2 bit assignments

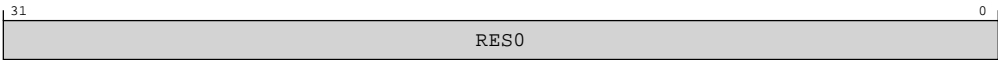


Table B-444: DBROM_DEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0xFC0	None

This interface is accessible as follows:

RO

B.2.2.3.24 DBROM_DEVID1, DebugBlock ROM table Device Configuration Register 1

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
DBROM

Register offset
0xFC4

Access type
RO

Reset value

xxxx

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-223: EXT_DBROM_DEVID1 bit assignments

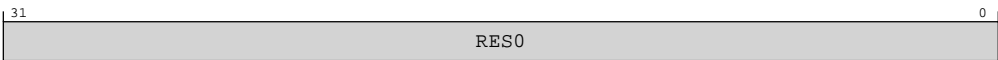


Table B-446: DBROM_DEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0xFC4	None

This interface is accessible as follows:

RO

B.2.2.3.25 DBROM_DEVID, DebugBlock ROM table Device Configuration Register

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFC8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-224: EXT_DBROM_DEVID bit assignments

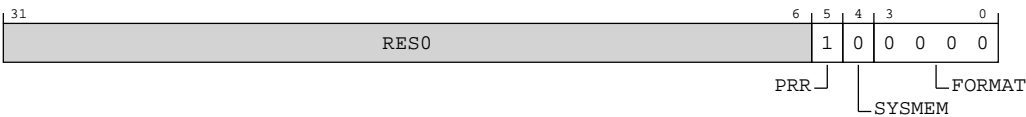


Table B-448: DBROM_DEVID bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	PRR	Power Request functionality included. 0b1 Power Request functionality included. ext-DBROM_PRIDR0 is implemented.	0b1

Bits	Name	Description	Reset
[4]	SYSMEM	System memory present. 0b0 System memory is not present on the bus.	0b0
[3:0]	FORMAT	ROM format. 0b0000 32-bit format 0.	0b0000

Accessibility

Component	Offset	Range
DBROM	0xFC8	None

This interface is accessible as follows:

RO

B.2.2.3.26 DBROM_DEVTYPE, DebugBlock 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

DBROM

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-225: EXT_DBROM_DEVTYPE bit assignments

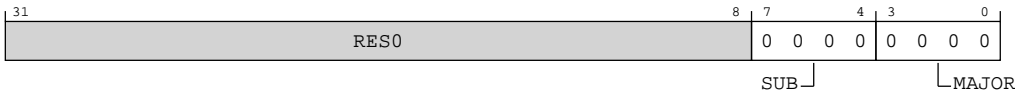


Table B-450: DBROM_DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SUB	Sub number 0b0000 Other, undefined.	0b0000
[3:0]	MAJOR	Major number 0b0000 Miscellaneous.	0b0000

Accessibility

Component	Offset	Range
DBROM	0xFCC	None

This interface is accessible as follows:

RO

B.2.2.3.27 DBROM_PIDR4, DebugBlock ROM table Peripheral Identification Register 4

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

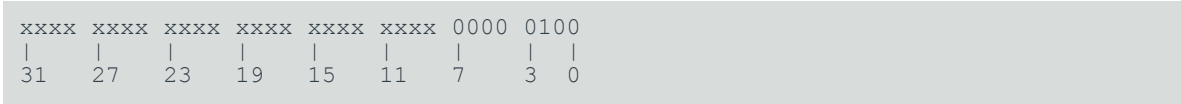
Register offset

0xFD0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-226: EXT_DBROM_PIDR4 bit assignments



Table B-452: DBROM_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
DBROM	0xFD0	None

This interface is accessible as follows:

RO

B.2.2.3.28 DBROM_PIDR5, DebugBlock ROM table Peripheral Identification Register 5

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
DBROM

Register offset
0xFD4

Access type
RO

Reset value

xxxx

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-227: EXT_DBROM_PIDR5 bit assignments

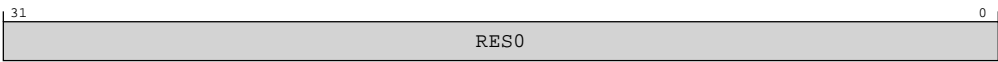


Table B-454: DBROM_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0xFD4	None

This interface is accessible as follows:

RO

B.2.2.3.29 DBROM_PIDR6, DebugBlock ROM table Peripheral Identification Register 6

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFD8

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-228: EXT_DBROM_PIDR6 bit assignments

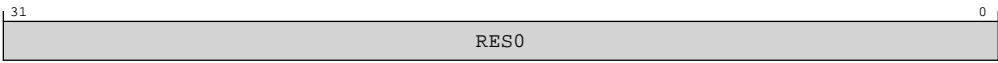


Table B-456: DBROM_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0xFD8	None

This interface is accessible as follows:

RO

B.2.2.3.30 DBROM_PIDR7, DebugBlock ROM table Peripheral Identification Register 7

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFDC

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-229: EXT_DBROM_PIDR7 bit assignments

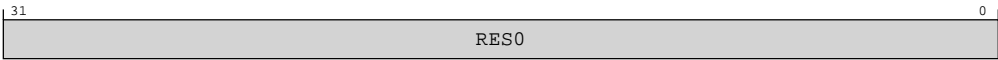


Table B-458: DBROM_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
DBROM	0xFDC	None

This interface is accessible as follows:

RO

B.2.2.3.31 DBROM_PIDR0, DebugBlock ROM table Peripheral Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFE0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-230: EXT_DBROM_PIDR0 bit assignments

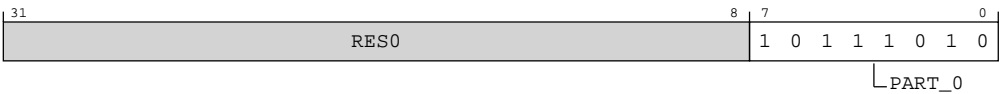


Table B-460: DBROM_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:0]	PART_0	Part number bits [7:0]. 0b10111010 Cortex-R82/Cortex-R82AE DebugBlock ROM table. Bits [7:0] of part number 0x4BA.	0xBA

Accessibility

Component	Offset	Range
DBROM	0xFE0	None

This interface is accessible as follows:

RO

B.2.2.3.32 DBROM_PIDR1, DebugBlock ROM table Peripheral Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFE4

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1011	0100
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-231: EXT_DBROM_PIDR1 bit assignments

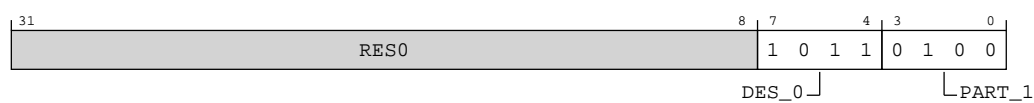


Table B-462: DBROM_PIDR1 bit descriptions

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]. 0b0100 Cortex-R82/Cortex-R82AE DebugBlock ROM table. Bits [11:8] of part number 0x4BA.	0b0100

Accessibility

Component	Offset	Range
DBROM	0xFE4	None

This interface is accessible as follows:

RO

B.2.2.3.33 DBROM_PIDR2, DebugBlock ROM table Peripheral Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFE8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-232: EXT_DBROM_PIDR2 bit assignments

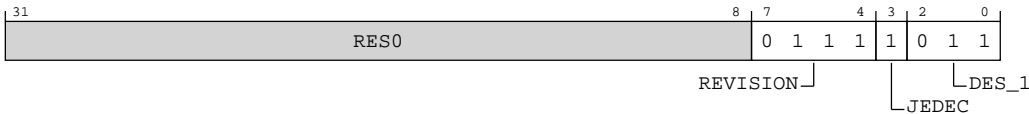


Table B-464: DBROM_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Range
DBROM	0xFE8	None

This interface is accessible as follows:

RO

B.2.2.3.34 DBROM_PIDR3, DebugBlock ROM table Peripheral Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

Register offset

0xFEC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-233: EXT_DBROM_PIDR3 bit assignments

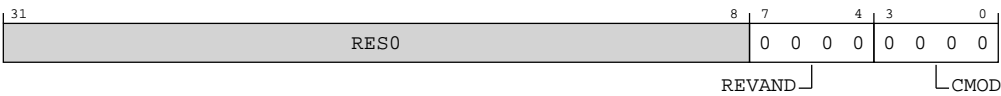


Table B-466: DBROM_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Minor errata fixes. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer Modified. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
DBROM	0xFEC	None

This interface is accessible as follows:

RO

B.2.2.3.35 DBROM_CIDR0, DebugBlock ROM table Component Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-234: EXT_DBROM_CIDR0 bit assignments



Table B-468: DBROM_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
DBROM	0xFF0	None

This interface is accessible as follows:

RO

B.2.2.3.36 DBROM_CIDR1, DebugBlock ROM table Component Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-235: EXT_DBROM_CIDR1 bit assignments

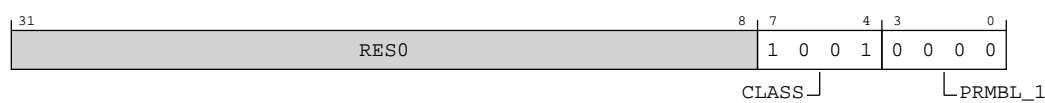


Table B-470: DBROM_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. 0b1001 CoreSight component.	0b1001
[3:0]	PRMBL_1	CoreSight component identification preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
DBROM	0xFF4	None

This interface is accessible as follows:

RO

B.2.2.3.37 DBROM_CIDR2, DebugBlock ROM table Component Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

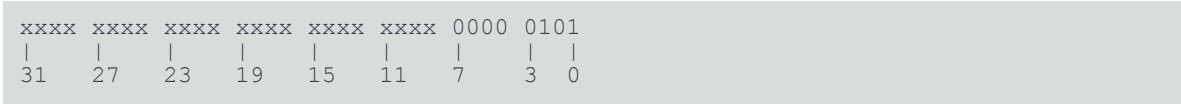
Register offset

0xFF8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-236: EXT_DBROM_CIDR2 bit assignments



Table B-472: DBROM_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	CoreSight component identification preamble. 0b000000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
DBROM	0xFF8	None

This interface is accessible as follows:

RO

B.2.2.3.38 DBROM_CIDR3, DebugBlock ROM table Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

DBROM

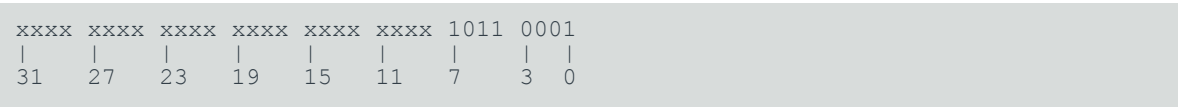
Register offset

0xFFC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-237: EXT_DBROM_CIDR3 bit assignments



Table B-474: DBROM_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
DBROM	0xFFC	None

This interface is accessible as follows:

RO

B.2.2.4 External CLUSTERROM register description

This section includes the register descriptions for all memory-mapped CLUSTERROM registers that are accessed for the cluster.

B.2.2.4.1 CLUSTERROM_ROMENTRY0, Cluster ROM table Entry 0

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x000

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	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-238: EXT_CLUSTERROM_ROMENTRY0 bit assignments

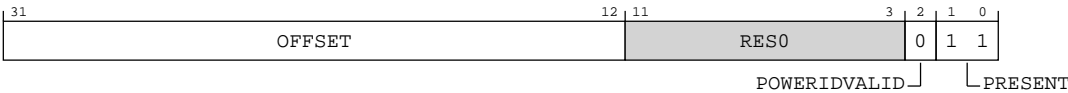


Table B-476: CLUSTERROM_ROMENTRY0 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000000001 Cluster PMU at address 0x0_2000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000000000010000 Cluster PMU at address 0x21_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Accessibility

Component	Offset	Range
CLUSTERROM	0x000	None

This interface is accessible as follows:

RO

B.2.2.4.2 CLUSTERROM_ROMENTRY1, Cluster ROM table Entry 1

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

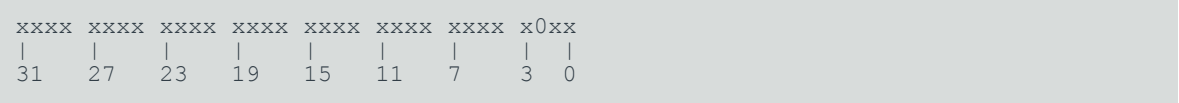
Register offset

0x004

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-239: EXT_CLUSTERROM_ROMENTRY1 bit assignments



Table B-478: CLUSTERROM_ROMENTRY1 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000000010 Cluster ELA at address 0x0_3000.</p> <p>Otherwise</p> <p>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).</p> <p>0000000000000000100000 Cluster ELA at address 0x22_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0

Bits	Name	Description	Reset
[1:0]	PRESENT	When ELA == 1 Indicates whether an entry is present at this location in the ROM Table. 11 The ROM Entry is present. Otherwise Indicates whether an entry is present at this location in the ROM Table. 10 The ROM Entry is not present, but this ROM Entry is not the final one in the ROM Table.	xx

Accessibility

Component	Offset	Range
CLUSTERROM	0x004	None

This interface is accessible as follows:

RO

B.2.2.4.3 CLUSTERROM_ROMENTRY2, Cluster ROM table Entry 2

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x008

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxx0	0000	x111
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-240: EXT_CLUSTERROM_ROMENTRY2 bit assignments

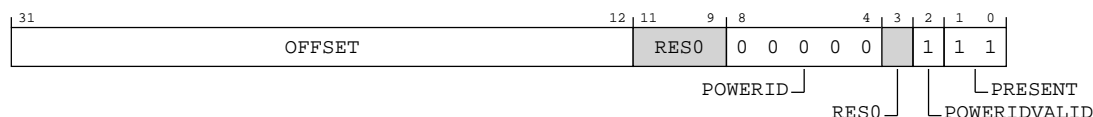


Table B-480: CLUSTERROM_ROMENTRY2 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>0000000000000000100 Core 0 ROM table at address 0x0_5000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000011000000000 Core 0 ROM table at address 0x80_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00000 PDCPU0 power domain.</p>	0b00000
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1 The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Accessibility

Component	Offset	Range
CLUSTERROM	0x008	None

This interface is accessible as follows:

RO

B.2.2.4.4 CLUSTERROM_ROMENTRY3, Cluster ROM table Entry 3

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x00C

Access type

RO

Reset value

When NUM_CORES >= 2

xxxx xxxx xxxx xxxx xxx0 0001 x111
xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 2

Figure B-241: EXT_CLUSTERROM_ROMENTRY3 bit assignments

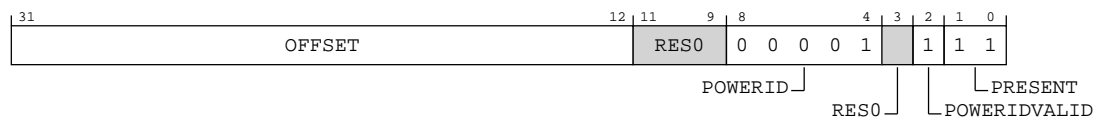


Table B-482: CLUSTERROM_ROMENTRY3 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000001010 Core 1 ROM table at address 0x0_B000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000011100000000 Core 1 ROM table at address 0x90_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00001 PDCPU1 power domain.</p>	0b00001
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1 The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-242: EXT_CLUSTERROM_ROMENTRY3 bit assignments**Table B-483: CLUSTERROM_ROMENTRY3 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0x00C	None

This interface is accessible as follows:

RO

B.2.2.4.5 CLUSTERROM_ROMENTRY4, Cluster ROM table Entry 4

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x010

Access type

RO

Reset value

When NUM_CORES >= 3

xxxx xxxx xxxx xxxx xxxx xxx0 0010 x111

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

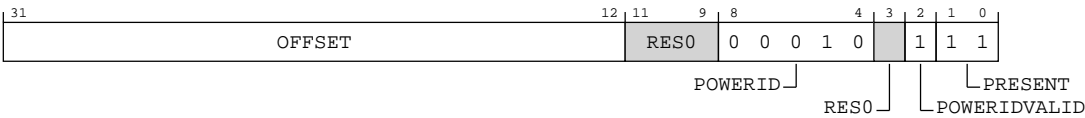


Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 3

Figure B-243: EXT_CLUSTERROM_ROMENTRY4 bit assignments



B.2.2.4.6 CLUSTERROM_ROMENTRY5, Cluster ROM table Entry 5

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x014

Access type

RO

Reset value

When NUM_CORES >= 4

xxxx xxxx xxxx xxxx xxxx xxx0 0011 x111

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 4

Figure B-245: EXT_CLUSTERROM_ROMENTRY5 bit assignments

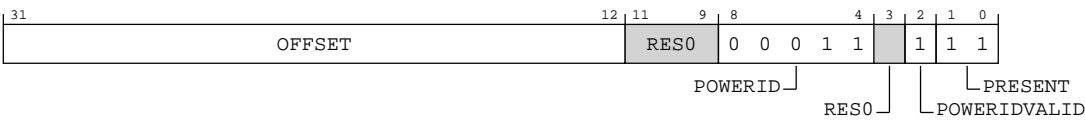


Table B-488: CLUSTERROM_ROMENTRY5 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>000000000000000010110 Core 3 ROM table at address 0x1_7000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000100100000000 Core 3 ROM table at address 0xB0_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00011 PDCPU3 power domain.</p>	0b00011
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1 The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-246: EXT_CLUSTERROM_ROMENTRY5 bit assignments**Table B-489: CLUSTERROM_ROMENTRY5 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0x014	None

This interface is accessible as follows:

RO

B.2.2.4.7 CLUSTERROM_ROMENTRY6, Cluster ROM table Entry 6

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x018

Access type

RO

Reset value

When NUM_CORES >= 5

xxxx xxxx xxxx xxxx xxxx xxx0 0100 x111

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 5

Figure B-247: EXT_CLUSTERROM_ROMENTRY6 bit assignments

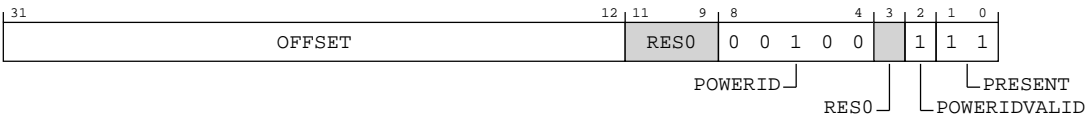
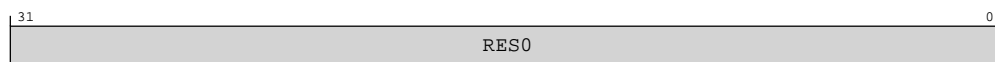


Table B-491: CLUSTERROM_ROMENTRY6 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>000000000000000011100 Core 4 ROM table at address 0x1_D000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000101000000000 Core 4 ROM table at address 0xC0_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00100 PDCPU4 power domain.</p>	0b00100
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1 The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-248: EXT_CLUSTERROM_ROMENTRY6 bit assignments**Table B-492: CLUSTERROM_ROMENTRY6 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0x018	None

This interface is accessible as follows:

RO

B.2.2.4.8 CLUSTERROM_ROMENTRY7, Cluster ROM table Entry 7

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x01C

Access type

RO

Reset value

When NUM_CORES >= 6

xxxx xxxx xxxx xxxx xxxx xxx0 0101 x111
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 6

Figure B-249: EXT_CLUSTERROM_ROMENTRY7 bit assignments

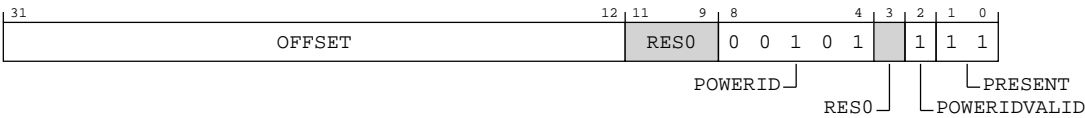


Table B-494: CLUSTERROM_ROMENTRY7 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000100010 Core 5 ROM table at address 0x2_3000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000101100000000 Core 5 ROM table at address 0xD0_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00101 PDCPU5 power domain.</p>	0b00101
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1 The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-250: EXT_CLUSTERROM_ROMENTRY7 bit assignments**Table B-495: CLUSTERROM_ROMENTRY7 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0x01C	None

This interface is accessible as follows:

RO

B.2.2.4.9 CLUSTERROM_ROMENTRY8, Cluster ROM table Entry 8

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x020

Access type

RO

Reset value

When NUM_CORES >= 7

xxxx xxxx xxxx xxxx xxxx xxx0 0110 x111

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 7

Figure B-251: EXT_CLUSTERROM_ROMENTRY8 bit assignments

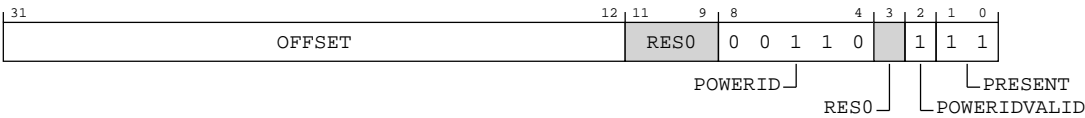


Table B-497: CLUSTERROM_ROMENTRY8 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000101000</p> <p>Core 6 ROM table at address 0x2_9000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000110000000000</p> <p>Core 6 ROM table at address 0xE0_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00110</p> <p>PD CPU6 power domain.</p>	0b00110
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1</p> <p>The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11</p> <p>The ROM Entry is present.</p>	0b11

Figure B-252: EXT_CLUSTERROM_ROMENTRY8 bit assignments

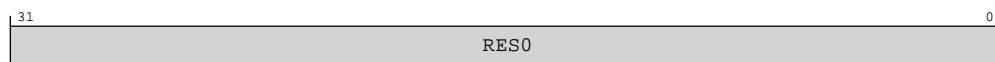


Table B-498: CLUSTERROM_ROMENTRY8 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0x020	None

This interface is accessible as follows:

RO

B.2.2.4.10 CLUSTERROM_ROMENTRY9, Cluster ROM table Entry 9

Provides the address offset for one CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0x024

Access type

RO

Reset value

When NUM_CORES == 8

xxxx xxxx xxxx xxxx xxxx xxx0 0111 x111

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES == 8

Figure B-253: EXT_CLUSTERROM_ROMENTRY9 bit assignments

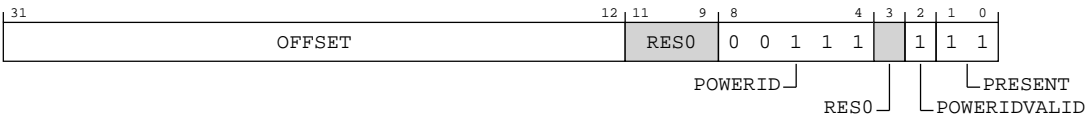
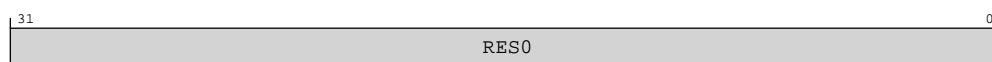


Table B-500: CLUSTERROM_ROMENTRY9 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000101110 Core 7 ROM table at address 0x2_F000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000110100000000 Core 7 ROM table at address 0xF0_0000.</p>	20{x}
[11:9]	RES0	Reserved	RES0
[8:4]	POWERID	<p>The power domain ID of the component.</p> <p>0b00111 PDCPU7 power domain.</p>	0b00111
[3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b1 The POWERID field provides a power domain ID.</p>	0b1
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Figure B-254: EXT_CLUSTERROM_ROMENTRY9 bit assignments**Table B-501: CLUSTERROM_ROMENTRY9 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0x024	None

This interface is accessible as follows:

RO

B.2.2.4.11 CLUSTERROM_DBGPCR0, Cluster ROM table Debug Power Control Register 0

Controls power requests for PDCPU0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

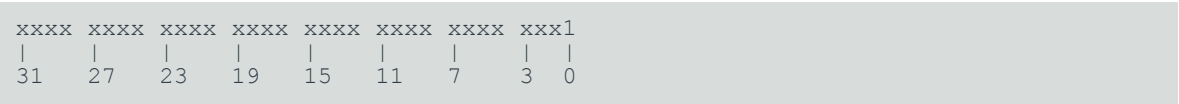
Register offset

0xA00

Access type

RW

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-255: EXT_CLUSTERROM_DBGPCR0 bit assignments



Table B-503: CLUSTERROM_DBGPCR0 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU0. 0b1 Power is requested for PDCPU0.	x

Bits	Name	Description	Reset
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU0 is implemented.	0b1

Accessibility

Component	Offset	Range
CLUSTERROM	0xA00	None

This interface is accessible as follows:

RW

B.2.2.4.12 CLUSTERROM_DBGPCR1, Cluster ROM table Debug Power Control Register 1

Controls power requests for PDCPU1.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA04

Access type

RW

Reset value

When NUM_CORES >= 2

xxxx xxxx xxxx xxxx xxxx xxxx xxx1
xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 2

Figure B-256: EXT_CLUSTERROM_DBGPCR1 bit assignments

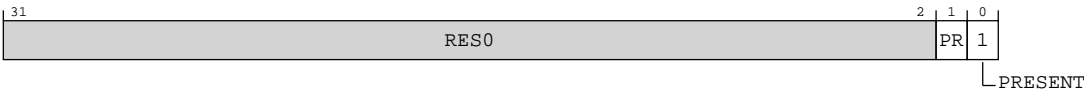


Table B-505: CLUSTERROM_DBGPCR1 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU1. 0b1 Power is requested for PDCPU1.	x
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU1 is implemented.	0b1

Figure B-257: EXT_CLUSTERROM_DBGPCR1 bit assignments

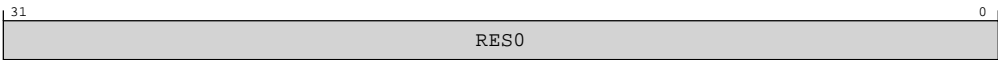


Table B-506: CLUSTERROM_DBGPCR1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA04	None

This interface is accessible as follows:

RW

B.2.2.4.13 CLUSTERROM_DBGPCR2, Cluster ROM table Debug Power Control Register 2

Controls power requests for PDCPU2.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA08

Access type

RW

Reset value

When NUM_CORES >= 3

xxxx xxxx xxxx xxxx xxxx xxxx xxx1
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 3

Figure B-258: EXT_CLUSTERROM_DBGPCR2 bit assignments

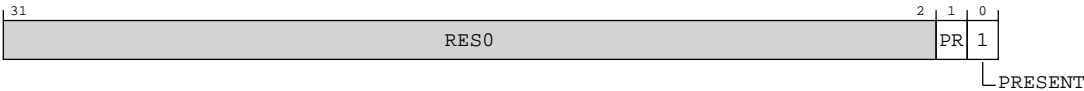


Table B-508: CLUSTERROM_DBGPCR2 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU2. 0b1 Power is requested for PDCPU2.	x
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU2 is implemented.	0b1

Figure B-259: EXT_CLUSTERROM_DBGPCR2 bit assignments

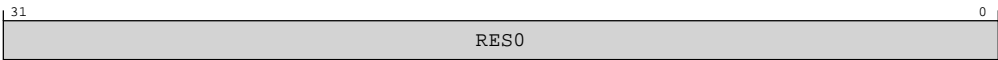


Table B-509: CLUSTERROM_DBGPCR2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA08	None

This interface is accessible as follows:

RW

B.2.2.4.14 CLUSTERROM_DBGPCR3, Cluster ROM table Debug Power Control Register 3

Controls power requests for PDCPU3.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA0C

Access type

RW

Reset value

When NUM_CORES >= 4

xxxx xxxx xxxx xxxx xxxx xxxx xxx1
xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 4

Figure B-260: EXT_CLUSTERROM_DBGPCR3 bit assignments

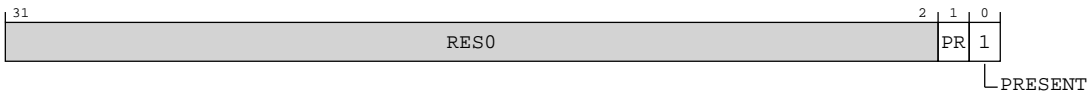


Table B-511: CLUSTERROM_DBGPCR3 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU3. 0b1 Power is requested for PDCPU3.	x
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU3 is implemented.	0b1

Figure B-261: EXT_CLUSTERROM_DBGPCR3 bit assignments

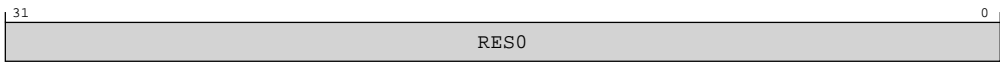


Table B-512: CLUSTERROM_DBGPCR3 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA0C	None

This interface is accessible as follows:

RW

B.2.2.4.15 CLUSTERROM_DBGPCR4, Cluster ROM table Debug Power Control Register 4

Controls power requests for PDCPU4.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA10

Access type

RW

Reset value

When NUM_CORES >= 5

xxxx xxxx xxxx xxxx xxxx xxxx xxx1
xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 5

Figure B-262: EXT_CLUSTERROM_DBGPCR4 bit assignments

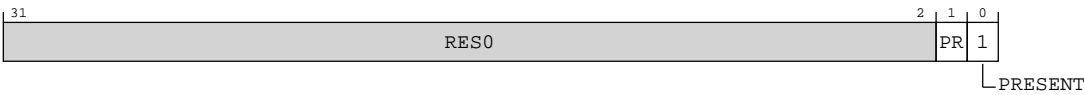
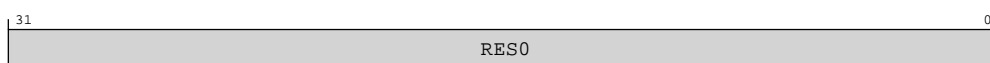


Table B-514: CLUSTERROM_DBGPCR4 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU4. 0b1 Power is requested for PDCPU4.	x
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU4 is implemented.	0b1

Figure B-263: EXT_CLUSTERROM_DBGPCR4 bit assignments**Table B-515: CLUSTERROM_DBGPCR4 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA10	None

This interface is accessible as follows:

RW

B.2.2.4.16 CLUSTERROM_DBGPCR5, Cluster ROM table Debug Power Control Register 5

Controls power requests for PDCPU5.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA14

Access type
RW

Reset value

When NUM_CORES >= 6

XXXX XXXX XXXX XXXX XXXX XXXX XXX1

XXXX XXXX XXXX XXXX XXXX XXXX XXXX



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 6

Figure B-264: EXT_CLUSTERROM_DBGPCR5 bit assignments



Table B-517: CLUSTERROM_DBGPCR5 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU5. 0b1 Power is requested for PDCPU5.	x
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU5 is implemented.	0b1

Figure B-265: EXT_CLUSTERROM_DBGPCR5 bit assignments

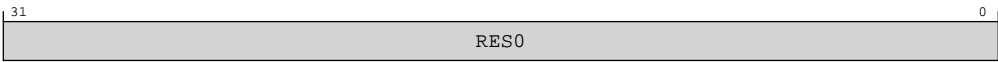


Table B-518: CLUSTERROM_DBGPCR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA14	None

This interface is accessible as follows:

RW

B.2.2.4.17 CLUSTERROM_DBGPCR6, Cluster ROM table Debug Power Control Register 6

Controls power requests for PDCPU6.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA18

Access type

RW

Reset value

When NUM_CORES >= 7

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxx1
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES >= 7

Figure B-266: EXT_CLUSTERROM_DBGPCR6 bit assignments

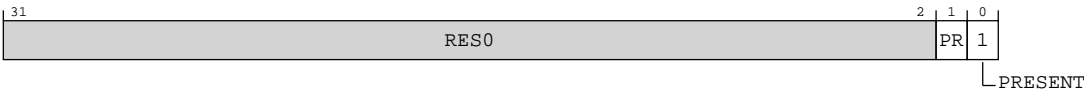


Table B-520: CLUSTERROM_DBGPCR6 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU6. 0b1 Power is requested for PDCPU6.	x
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU6 is implemented.	0b1

Figure B-267: EXT_CLUSTERROM_DBGPCR6 bit assignments

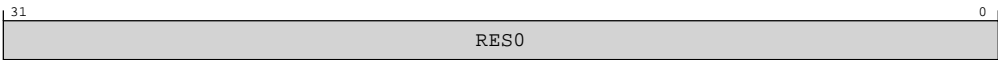


Table B-521: CLUSTERROM_DBGPCR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA18	None

This interface is accessible as follows:

RW

B.2.2.4.18 CLUSTERROM_DBGPCR7, Cluster ROM table Debug Power Control Register 7

Controls power requests for PDCPU7.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA1C

Access type

RW

Reset value

When NUM_CORES == 8

xxxx xxxx xxxx xxxx xxxx xxxx xxx1
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When NUM_CORES == 8

Figure B-268: EXT_CLUSTERROM_DBGPCR7 bit assignments

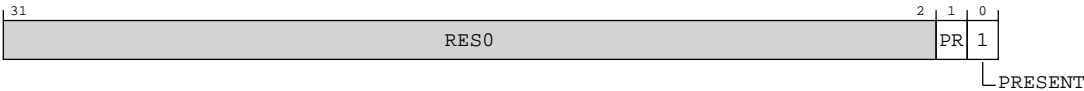


Table B-523: CLUSTERROM_DBGPCR7 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PR	Power Request. 0b0 Power is not requested for PDCPU7. 0b1 Power is requested for PDCPU7.	x
[0]	PRESENT	Power request implemented. 0b1 Power request for PDCPU7 is implemented.	0b1

Figure B-269: EXT_CLUSTERROM_DBGPCR7 bit assignments

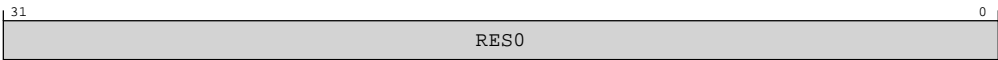


Table B-524: CLUSTERROM_DBGPCR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA1C	None

This interface is accessible as follows:

RW

B.2.2.4.19 CLUSTERROM_DBGPSR0, Cluster ROM table Debug Power Status Register 0

Indicates the power status for PDCPU0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

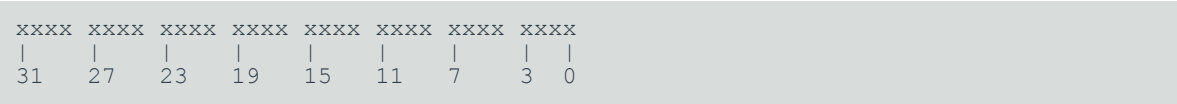
Register offset

0xA80

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-270: EXT_CLUSTERROM_DBGPSR0 bit assignments

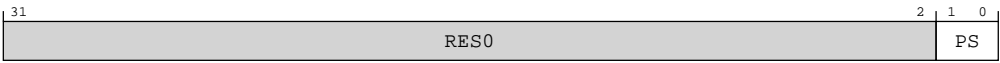


Table B-526: CLUSTERROM_DBGPSR0 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	PS	Power Status. 0b00 PDCPU0 might not be powered. 0b01 PDCPU0 is powered.	xx

Accessibility

Component	Offset	Range
CLUSTERROM	0xA80	None

This interface is accessible as follows:

RO

B.2.2.4.20 CLUSTERROM_DBGPSR1, Cluster ROM table Debug Power Status Register 1

Indicates the power status for PDCPU1.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA84

Access type

RO

Reset value

When NUM_CORES >= 2

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 NUM_CORES >= 2

Figure B-271: EXT_CLUSTERROM_DBGPSR1 bit assignments

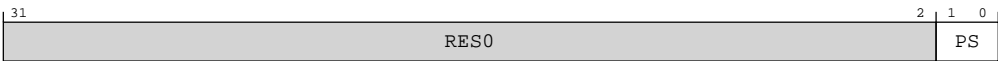


Table B-528: CLUSTERROM_DBGPSR1 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	PS	Power Status. 0b00 PDCPU1 might not be powered. 0b01 PDCPU1 is powered.	xx

Figure B-272: EXT_CLUSTERROM_DBGPSR1 bit assignments

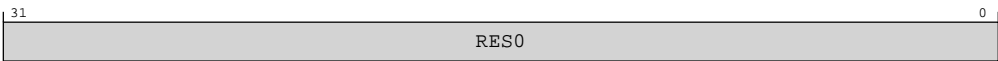


Table B-529: CLUSTERROM_DBGPSR1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA84	None

This interface is accessible as follows:

RO

B.2.2.4.21 CLUSTERROM_DBGPSR2, Cluster ROM table Debug Power Status Register 2

Indicates the power status for PDCPU2.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA88

Access type

RO

Reset value

When NUM_CORES >= 3

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 NUM_CORES >= 3

Figure B-273: EXT_CLUSTERROM_DBGPSR2 bit assignments

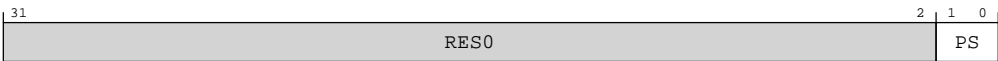


Table B-531: CLUSTERROM_DBGPSR2 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1:0]	PS	Power Status. 0b00 PDCPU2 might not be powered. 0b01 PDCPU2 is powered.	xx

Figure B-274: EXT_CLUSTERROM_DBGPSR2 bit assignments

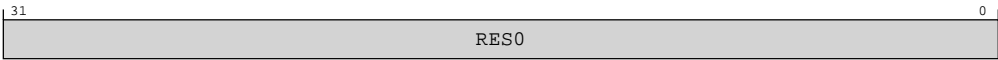


Table B-532: CLUSTERROM_DBGPSR2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA88	None

This interface is accessible as follows:

RO

B.2.2.4.22 CLUSTERROM_DBGPSR3, Cluster ROM table Debug Power Status Register 3

Indicates the power status for PDCPU3.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA8C

Access type

RO

Reset value

When NUM_CORES >= 4

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 NUM_CORES >= 4

Figure B-275: EXT_CLUSTERROM_DBGPSR3 bit assignments

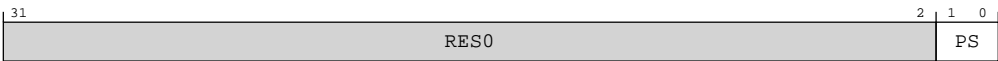


Table B-534: CLUSTERROM_DBGPSR3 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	PS	Power Status. 0b00 PDCPU3 might not be powered. 0b01 PDCPU3 is powered.	xx

Figure B-276: EXT_CLUSTERROM_DBGPSR3 bit assignments



Table B-535: CLUSTERROM_DBGPSR3 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA8C	None

This interface is accessible as follows:

RO

B.2.2.4.23 CLUSTERROM_DBGPSR4, Cluster ROM table Debug Power Status Register 4

Indicates the power status for PDCPU4.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA90

Access type

RO

Reset value

When NUM_CORES >= 5

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 NUM_CORES >= 5

Figure B-277: EXT_CLUSTERROM_DBGPSR4 bit assignments

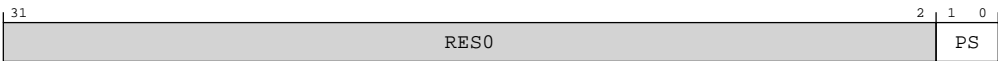


Table B-537: CLUSTERROM_DBGPSR4 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1:0]	PS	Power Status. 0b00 PDCPU4 might not be powered. 0b01 PDCPU4 is powered.	xx

Figure B-278: EXT_CLUSTERROM_DBGPSR4 bit assignments

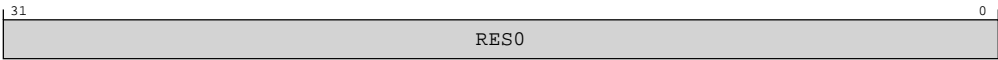


Table B-538: CLUSTERROM_DBGPSR4 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA90	None

This interface is accessible as follows:

RO

B.2.2.4.24 CLUSTERROM_DBGPSR5, Cluster ROM table Debug Power Status Register 5

Indicates the power status for PDCPU5.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA94

Access type

RO

Reset value

When NUM_CORES >= 6

XXXX XXXX XXXX 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 NUM_CORES >= 6

Figure B-279: EXT_CLUSTERROM_DBGPSR5 bit assignments

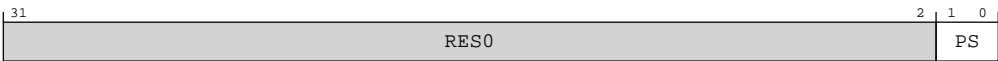


Table B-540: CLUSTERROM_DBGPSR5 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	PS	Power Status. 0b00 PDCPU5 might not be powered. 0b01 PDCPU5 is powered.	xx

Figure B-280: EXT_CLUSTERROM_DBGPSR5 bit assignments

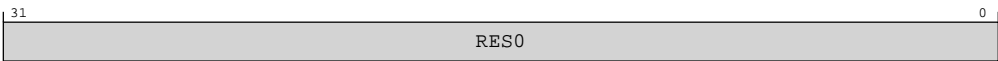


Table B-541: CLUSTERROM_DBGPSR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA94	None

This interface is accessible as follows:

RO

B.2.2.4.25 CLUSTERROM_DBGPSR6, Cluster ROM table Debug Power Status Register 6

Indicates the power status for PDCPU6.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA98

Access type

RO

Reset value

When NUM_CORES >= 7

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 NUM_CORES >= 7

Figure B-281: EXT_CLUSTERROM_DBGPSR6 bit assignments

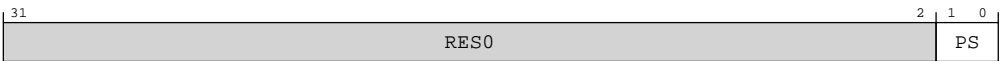


Table B-543: CLUSTERROM_DBGPSR6 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1:0]	PS	Power Status. 0b00 PDCPU6 might not be powered. 0b01 PDCPU6 is powered.	xx

Figure B-282: EXT_CLUSTERROM_DBGPSR6 bit assignments

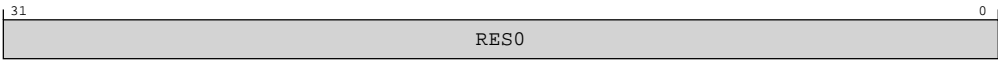


Table B-544: CLUSTERROM_DBGPSR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA98	None

This interface is accessible as follows:

RO

B.2.2.4.26 CLUSTERROM_DBGPSR7, Cluster ROM table Debug Power Status Register 7

Indicates the power status for PDCPU7.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xA9C

Access type

RO

Reset value

When NUM_CORES == 8

XXXX XXXX XXXX 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 NUM_CORES == 8

Figure B-283: EXT_CLUSTERROM_DBGPSR7 bit assignments

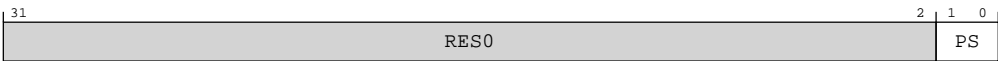


Table B-546: CLUSTERROM_DBGPSR7 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1:0]	PS	Power Status. 0b00 PDCPU7 might not be powered. 0b01 PDCPU7 is powered.	xx

Figure B-284: EXT_CLUSTERROM_DBGPSR7 bit assignments

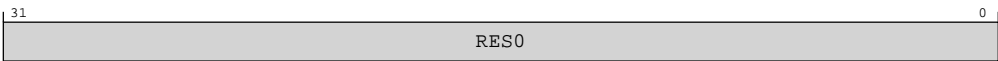


Table B-547: CLUSTERROM_DBGPSR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xA9C	None

This interface is accessible as follows:

RO

B.2.2.4.27 CLUSTERROM_PRIDR0, Cluster ROM table Power Request ID Register 0

Indicates the features of the power request functionality.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

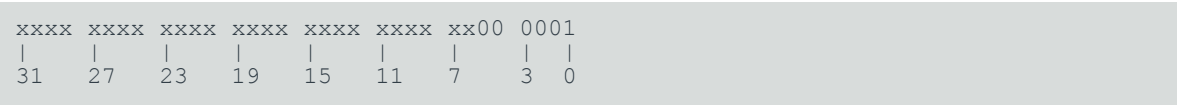
Register offset

0xC00

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-285: EXT_CLUSTERROM_PRIDR0 bit assignments

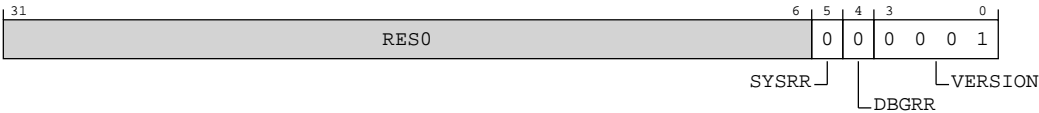


Table B-549: CLUSTERROM_PRIDR0 bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	SYSRR	System reset request functionality present. 0b0 The system reset request functionality is not implemented.	0b0

Bits	Name	Description	Reset
[4]	DBGRR	Debug reset request functionality present. 0b0 The debug reset request functionality is not implemented.	0b0
[3:0]	VERSION	Version of the power request functionality. 0b0001 The power request functionality version 0, and the per-core controls for power requests (e.g. ext-CLUSTERROM_DBGPCRO and ext-CLUSTERROM_DBGPSRO), are implemented.	0b0001

Accessibility

Component	Offset	Range
CLUSTERROM	0xC00	None

This interface is accessible as follows:

RO

B.2.2.4.28 CLUSTERROM_ITCTRL, Cluster ROM table Integration Mode Control Register

No functional/integration mode switching implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xF00

Access type

RW

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-286: EXT_CLUSTERROM_ITCTRL bit assignments

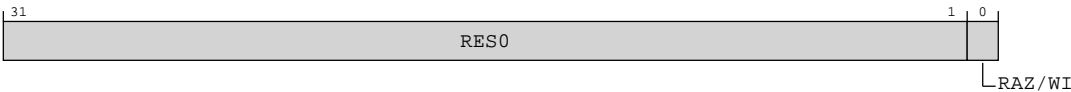


Table B-551: CLUSTERROM_ITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERROM	0xF00	None

This interface is accessible as follows:

RW

B.2.2.4.29 CLUSTERROM_CLAIMSET, Cluster ROM table Claim Tag Set Register

No claim tags implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFA0

Access type

RW

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-287: EXT_CLUSTERROM_CLAIMSET bit assignments

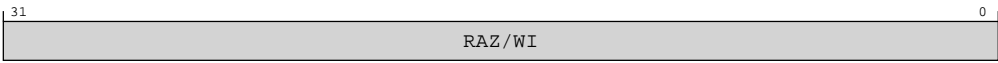


Table B-553: CLUSTERROM_CLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERROM	0xFA0	None

This interface is accessible as follows:

RW

B.2.2.4.30 CLUSTERROM_CLAIMCLR, Cluster ROM table Claim Tag Clear Register

No claim tags implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFA4

Access type

RW

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-288: EXT_CLUSTERROM_CLAIMCLR bit assignments

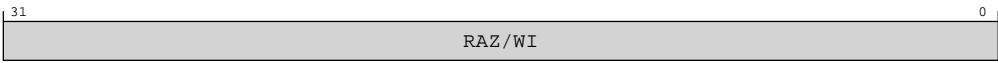


Table B-555: CLUSTERROM_CLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERROM	0xFA4	None

This interface is accessible as follows:

RW

B.2.2.4.31 CLUSTERROM_DEVAFF0, Cluster ROM table Device Affinity Register 0

Enables a debugger to determine whether two components have an affinity with each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFA8

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-289: EXT_CLUSTERROM_DEVAFF0 bit assignments

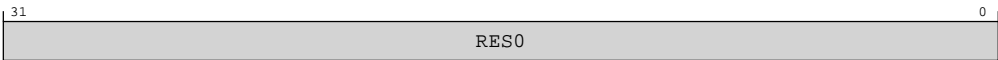


Table B-557: CLUSTERROM_DEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xFA8	None

This interface is accessible as follows:

RO

B.2.2.4.32 CLUSTERROM_DEVAFF1, Cluster ROM table Device Affinity Register 1

Enables a debugger to determine whether two components have an affinity with each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

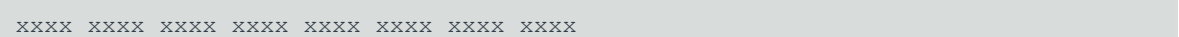
Register offset

0xFAC

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-290: EXT_CLUSTERROM_DEVAFF1 bit assignments

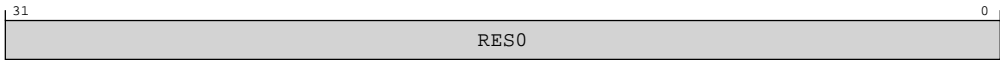


Table B-559: CLUSTERROM_DEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xFAC	None

This interface is accessible as follows:

RO

B.2.2.4.33 CLUSTERROM_LAR, Cluster ROM table Software Lock Access Register

No software locking implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFB0

Access type

RESERVEDW

Reset value
0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions
Figure B-291: EXT_CLUSTERROM_LAR bit assignments

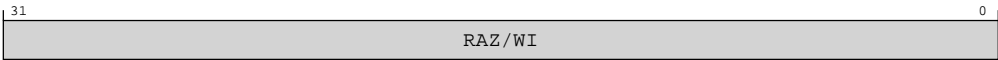


Table B-561: CLUSTERROM_LAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERROM	0xFB0	None

This interface is accessible as follows:

WO

B.2.2.4.34 CLUSTERROM_LSR, Cluster ROM table Software Lock Status Register

No software locking implemented for ROM tables.

Configurations
This register is available in all configurations.

Attributes

- Width**
32
- Component**
CLUSTERROM
- Register offset**
0xFB4
- Access type**
RO
- Reset value**
0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-292: EXT_CLUSTERROM_LSR bit assignments

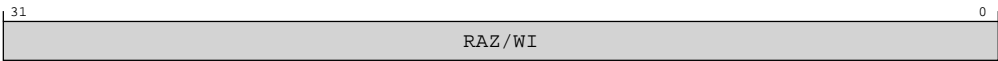


Table B-563: CLUSTERROM_LSR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
CLUSTERROM	0xFB4	None

This interface is accessible as follows:

RO

B.2.2.4.35 CLUSTERROM_AUTHSTATUS, Cluster 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

CLUSTERROM

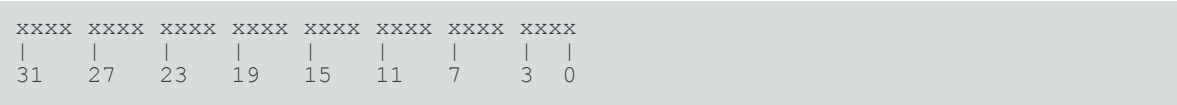
Register offset

0xFB8

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-293: EXT_CLUSTERROM_AUTHSTATUS bit assignments



Table B-565: CLUSTERROM_AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure Non-invasive Debug. ExternalSecureNoninvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field.	xx
[5:4]	SID	Secure Invasive Debug. 0b10 Secure invasive debug disabled. ExternalSecureInvasiveDebugEnabled() == FALSE. 0b11 Secure invasive debug enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.	xx
[3:2]	NSNID	Non-secure Non-invasive Debug. ExternalNoninvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field.	xx
[1:0]	NSID	Non-secure Invasive Debug. ExternalInvasiveDebugEnabled() == ExternalSecureInvasiveDebugEnabled(). This field has the same value as the SID field.	xx

Accessibility

Component	Offset	Range
CLUSTERROM	0xFB8	None

This interface is accessible as follows:

RO

B.2.2.4.36 CLUSTERROM_DEVARCH, Cluster 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

CLUSTERROM

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 0000 0000 1010 1111 0111

Bit descriptions

Figure B-294: EXT_CLUSTERROM_DEVARCH bit assignments

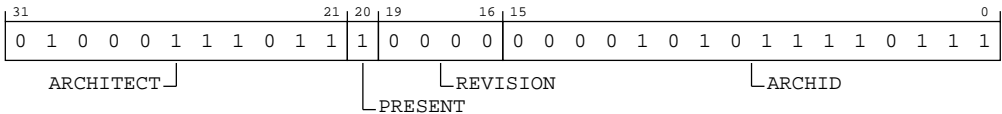


Table B-567: CLUSTERROM_DEVARCH bit descriptions

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-CLUSTERROM_DEVTYPE and ext-CLUSTERROM_DEVID to determine further information about the ROM Table.	0x0AF7

Accessibility

Component	Offset	Range
CLUSTERROM	0xFBC	None

This interface is accessible as follows:

RO

B.2.2.4.37 CLUSTERROM_DEVID2, Cluster ROM table Device Configuration Register 2

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

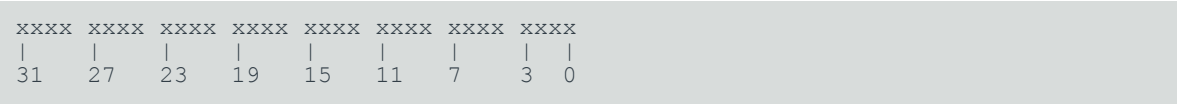
Register offset

0xFC0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-295: EXT_CLUSTERROM_DEVID2 bit assignments

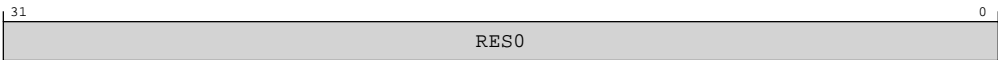


Table B-569: CLUSTERROM_DEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xFC0	None

This interface is accessible as follows:

RO

B.2.2.4.38 CLUSTERROM_DEVID1, Cluster ROM table Device Configuration Register 1

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFC4

Access type

RO

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-296: EXT_CLUSTERROM_DEVID1 bit assignments

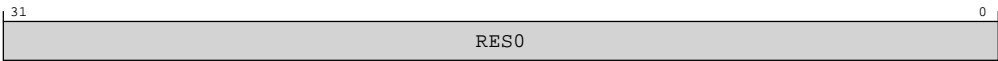


Table B-571: CLUSTERROM_DEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xFC4	None

This interface is accessible as follows:

RO

B.2.2.4.39 CLUSTERROM_DEVID, Cluster ROM table Device Configuration Register

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

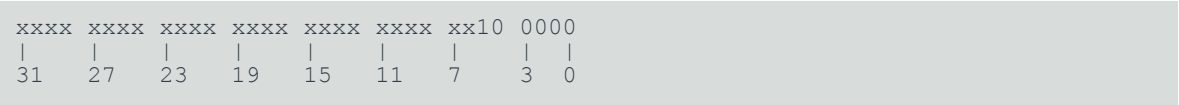
Register offset

0xFC8

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-297: EXT_CLUSTERROM_DEVID bit assignments

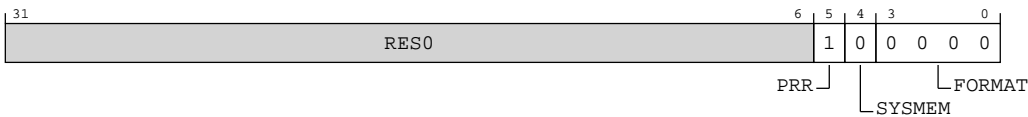


Table B-573: CLUSTERROM_DEVID bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	PRR	Power Request functionality included. 0b1 Power Request functionality included. ext-CLUSTERROM_PRIDR0 is implemented.	0b1
[4]	SYSMEM	System memory present. 0b0 System memory is not present on the bus.	0b0
[3:0]	FORMAT	ROM format. 0b0000 32-bit format 0.	0b0000

Accessibility

Component	Offset	Range
CLUSTERROM	0xFC8	None

This interface is accessible as follows:

RO

B.2.2.4.40 CLUSTERROM_DEVTYPE, Cluster 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

CLUSTERROM

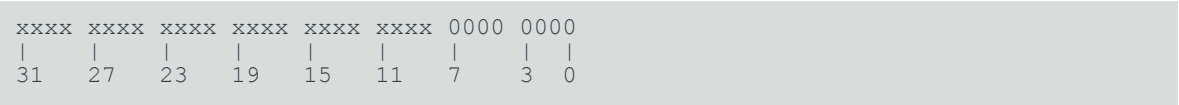
Register offset

0xFCC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-298: EXT_CLUSTERROM_DEVTYPE bit assignments



Table B-575: CLUSTERROM_DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SUB	Sub number 0b0000 Other, undefined.	0b0000
[3:0]	MAJOR	Major number 0b0000 Miscellaneous.	0b0000

Accessibility

Component	Offset	Range
CLUSTERROM	0xFCC	None

This interface is accessible as follows:

RO

B.2.2.4.41 CLUSTERROM_PIDR4, Cluster ROM table Peripheral Identification Register 4

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

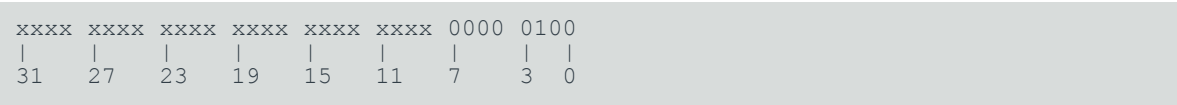
Register offset

0xFD0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-299: EXT_CLUSTERROM_PIDR4 bit assignments

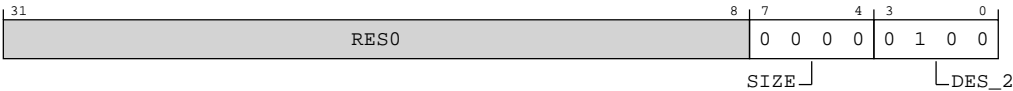


Table B-577: CLUSTERROM_PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	4KB count. 0b0000 The component uses a single 4KB block.	0b0000

Bits	Name	Description	Reset
[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
CLUSTERROM	0xFD0	None

This interface is accessible as follows:

RO

B.2.2.4.42 CLUSTERROM_PIDR5, Cluster ROM table Peripheral Identification Register 5

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFD4

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-300: EXT_CLUSTERROM_PIDR5 bit assignments

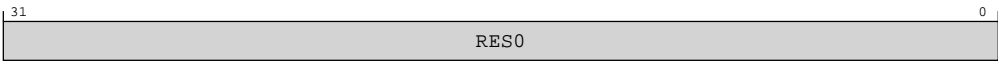


Table B-579: CLUSTERROM_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xFD4	None

This interface is accessible as follows:

RO

B.2.2.4.43 CLUSTERROM_PIDR6, Cluster ROM table Peripheral Identification Register 6

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFD8

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-301: EXT_CLUSTERROM_PIDR6 bit assignments

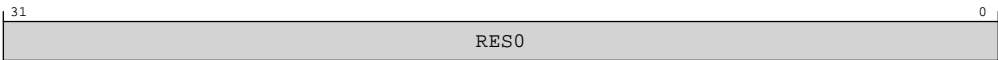


Table B-581: CLUSTERROM_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xFD8	None

This interface is accessible as follows:

RO

B.2.2.4.44 CLUSTERROM_PIDR7, Cluster ROM table Peripheral Identification Register 7

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

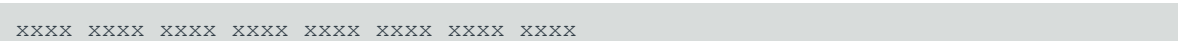
Register offset

0xFDC

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-302: EXT_CLUSTERROM_PIDR7 bit assignments

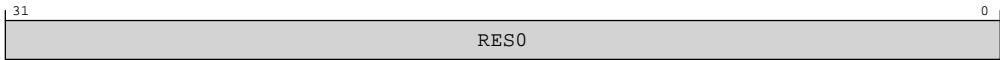


Table B-583: CLUSTERROM_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
CLUSTERROM	0xFDC	None

This interface is accessible as follows:

RO

B.2.2.4.45 CLUSTERROM_PIDR0, Cluster ROM table Peripheral Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

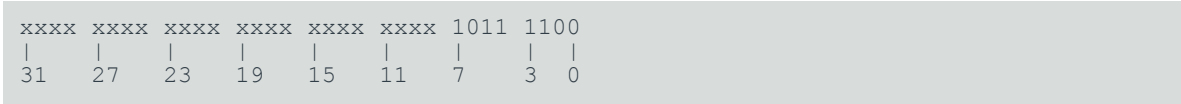
Register offset

0xFE0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-303: EXT_CLUSTERROM_PIDR0 bit assignments



Table B-585: CLUSTERROM_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b10111100 Cortex-R82 Cluster ROM table. Bits [7:0] of part number 0x4BC.	0xBC

Accessibility

Component	Offset	Range
CLUSTERROM	0xFE0	None

This interface is accessible as follows:

RO

B.2.2.4.46 CLUSTERROM_PIDR1, Cluster ROM table Peripheral Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

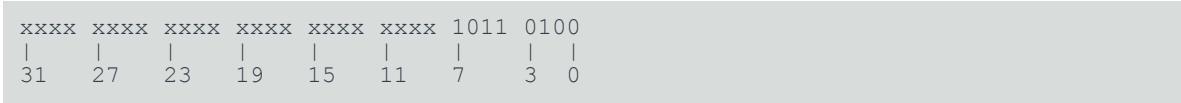
32

Component
CLUSTERROM

Register offset
0xFE4

Access type
RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions
Figure B-304: EXT_CLUSTERROM_PIDR1 bit assignments



Table B-587: CLUSTERROM_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]. 0b0100 Cortex-R82 Cluster ROM table. Bits [11:8] of part number 0x4BC.	0b0100

Accessibility

Component	Offset	Range
CLUSTERROM	0xFE4	None

This interface is accessible as follows:

RO

B.2.2.4.47 CLUSTERROM_PIDR2, Cluster ROM table Peripheral Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

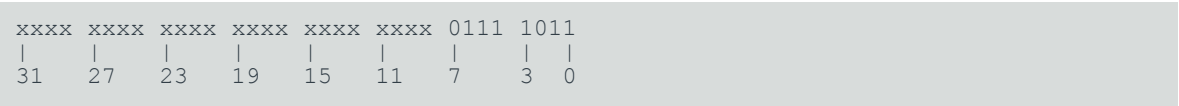
Register offset

0xFE8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-305: EXT_CLUSTERROM_PIDR2 bit assignments

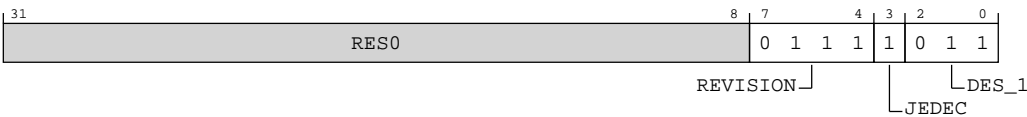


Table B-589: CLUSTERROM_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111

Bits	Name	Description	Reset
[3]	JEDEC	RAO . Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Range
CLUSTERROM	0xFE8	None

This interface is accessible as follows:

RO

B.2.2.4.48 CLUSTERROM_PIDR3, Cluster ROM table Peripheral Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

Register offset

0xFEC

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-306: EXT_CLUSTERROM_PIDR3 bit assignments

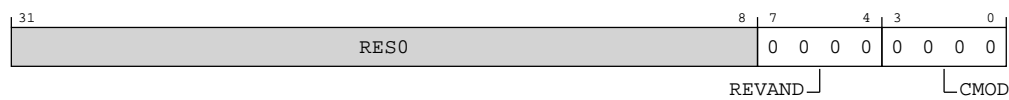


Table B-591: CLUSTERROM_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Minor errata fixes. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer Modified. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
CLUSTERROM	0xFEC	None

This interface is accessible as follows:

RO

B.2.2.4.49 CLUSTERROM_CIDR0, Cluster ROM table Component Identification Register 0

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

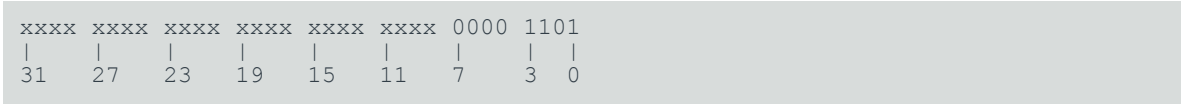
Register offset

0xFF0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-307: EXT_CLUSTERROM_CIDR0 bit assignments



Table B-593: CLUSTERROM_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
CLUSTERROM	0xFF0	None

This interface is accessible as follows:

RO

B.2.2.4.50 CLUSTERROM_CIDR1, Cluster ROM table Component Identification Register 1

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component
CLUSTERROM

Register offset
0xFF4

Access type
RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions
Figure B-308: EXT_CLUSTERROM_CIDR1 bit assignments



Table B-595: CLUSTERROM_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. 0b1001 CoreSight component.	0b1001
[3:0]	PRMBL_1	CoreSight component identification preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
CLUSTERROM	0xFF4	None

This interface is accessible as follows:

RO

B.2.2.4.51 CLUSTERROM_CIDR2, Cluster ROM table Component Identification Register 2

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

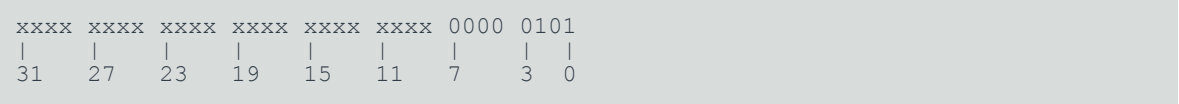
Register offset

0xFF8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-309: EXT_CLUSTERROM_CIDR2 bit assignments

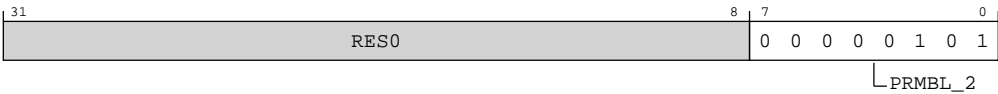


Table B-597: CLUSTERROM_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	CoreSight component identification preamble. 0b000000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
CLUSTERROM	0xFF8	None

This interface is accessible as follows:

RO

B.2.2.4.52 CLUSTERROM_CIDR3, Cluster ROM table Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CLUSTERROM

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-310: EXT_CLUSTERROM_CIDR3 bit assignments

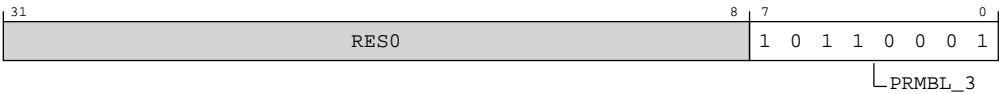


Table B-599: CLUSTERROM_CIDR3 bit descriptions

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
CLUSTERROM	0xFFC	None

This interface is accessible as follows:

RO

B.2.2.5 External COREROM register description

This section includes the register descriptions for all memory-mapped COREROM registers that are accessed for each core.

B.2.2.5.1 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

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x011
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-311: EXT_COREROM_ROMENTRY0 bit assignments

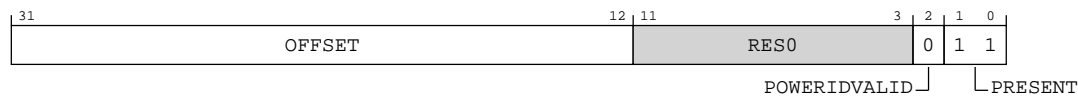


Table B-601: COREROM_ROMENTRY0 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	When DENSE_CS_ADDR_MAP == 1 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). 00000000000000000001 Core Debug at ROM table address + 0x1000. Otherwise 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). 0000000000000000010000 Core Debug at ROM table address + 0x1_0000.	20 {x}
[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	0x000	None

This interface is accessible as follows:

RO

B.2.2.5.2 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

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x011
31	27	23	19	15	11	7	3	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-312: EXT_COREROM_ROMENTRY1 bit assignments

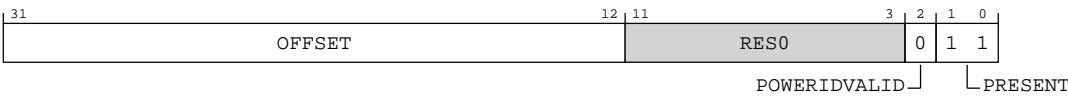


Table B-603: COREROM_ROMENTRY1 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>00000000000000000010 Core PMU at ROM table address + 0x2000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000000000100000 Core PMU at ROM table address + 0x2_0000.</p>	20 {x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0 A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11 The ROM Entry is present.</p>	0b11

Accessibility

Component	Offset	Range
COREROM	0x004	None

This interface is accessible as follows:

RO

B.2.2.5.3 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

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x011
31	27	23	19	15	11	7	3	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-313: EXT_COREROM_ROMENTRY2 bit assignments

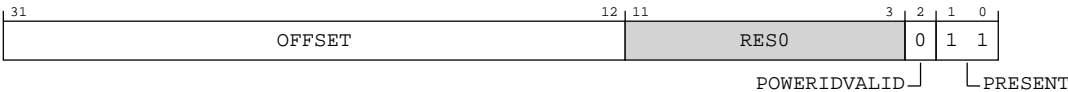


Table B-605: COREROM_ROMENTRY2 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	When DENSE_CS_ADDR_MAP == 1 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). 00000000000000000011 Core Trace at ROM table address + 0x3000. Otherwise 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). 00000000000000110000 Core Trace at ROM table address + 0x3_0000.	20 {x}
[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.2.2.5.4 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

When ELA == 1

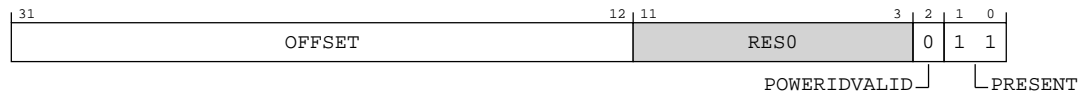
xxxx xxxx xxxx xxxx xxxx xxxx xxxx x011
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx



Where the reset reads xxxx, see individual bits.

Bit descriptions

When ELA == 1

Figure B-314: EXT_COREROM_ROMENTRY3 bit assignments**Table B-607: COREROM_ROMENTRY3 bit descriptions**

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>When DENSE_CS_ADDR_MAP == 1</p> <p>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).</p> <p>0000000000000000100</p> <p>Core ELA at ROM table address + 0x4000.</p> <p>Otherwise</p> <p>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).</p> <p>00000000000001000000</p> <p>Core ELA at ROM table address + 0x4_0000.</p>	20{x}
[11:3]	RES0	Reserved	RES0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p>0b0</p> <p>A power domain ID is not provided.</p>	0b0
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p>0b11</p> <p>The ROM Entry is present.</p>	0b11

Figure B-315: EXT_COREROM_ROMENTRY3 bit assignments**Table B-608: COREROM_ROMENTRY3 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0x00C	None

This interface is accessible as follows:

RO

B.2.2.5.5 COREROM_PRIDR0, Core ROM table Power Request ID Register 0

Indicates the features of the power request functionality.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

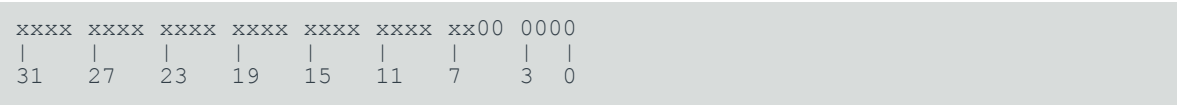
Register offset

0xC00

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-316: EXT_COREROM_PRIDR0 bit assignments

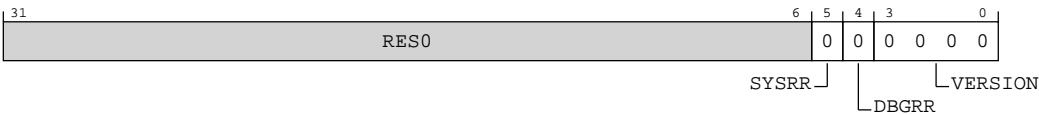


Table B-610: COREROM_PRIDR0 bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	SYSRR	System reset request functionality present. 0b0 The system reset request functionality is not implemented.	0b0

Bits	Name	Description	Reset
[4]	DBGRR	Debug reset request functionality present. 0b0 The debug reset request functionality is not implemented.	0b0
[3:0]	VERSION	Version of the power request functionality. 0b0000 The power request functionality is not implemented.	0b0000

Accessibility

Component	Offset	Range
COREROM	0xC00	None

This interface is accessible as follows:

RO

B.2.2.5.6 COREROM_ITCTRL, Core ROM table Integration Mode Control Register

No functional/integration mode switching implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xF00

Access type

RW

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-317: EXT_COREROM_ITCTRL bit assignments

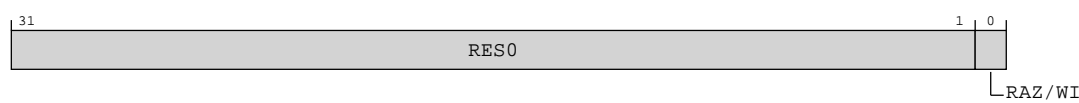


Table B-612: COREROM_ITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
COREROM	0xF00	None

This interface is accessible as follows:

RW

B.2.2.5.7 COREROM_CLAIMSET, Core ROM table Claim Tag Set Register

No claim tags implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFA0

Access type

RW

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-318: EXT_COREROM_CLAIMSET bit assignments

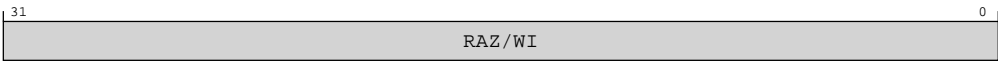


Table B-614: COREROM_CLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
COREROM	0xFA0	None

This interface is accessible as follows:

RW

B.2.2.5.8 COREROM_CLAIMCLR, Core ROM table Claim Tag Clear Register

No claim tags implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFA4

Access type

RW

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-319: EXT_COREROM_CLAIMCLR bit assignments

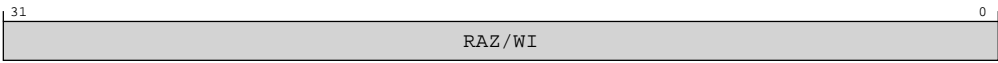


Table B-616: COREROM_CLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
COREROM	0xFA4	None

This interface is accessible as follows:

RW

B.2.2.5.9 COREROM_DEVAFF0, Core ROM table Device Affinity Register 0

Enables a debugger to determine whether two components have an affinity with each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

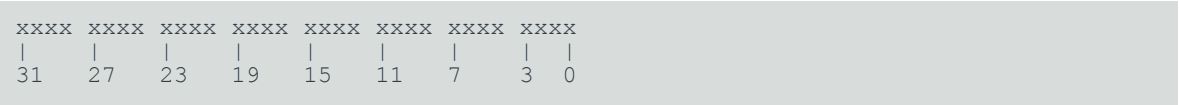
Register offset

0xFA8

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-320: EXT_COREROM_DEVAFF0 bit assignments

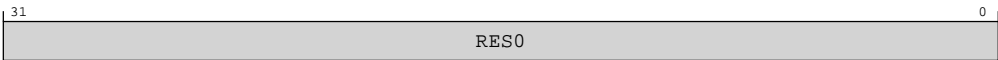


Table B-618: COREROM_DEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0xFA8	None

This interface is accessible as follows:

RO

B.2.2.5.10 COREROM_DEVAFF1, Core ROM table Device Affinity Register 1

Enables a debugger to determine whether two components have an affinity with each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

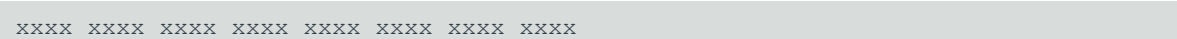
Register offset

0xFAC

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-321: EXT_COREROM_DEVAFF1 bit assignments

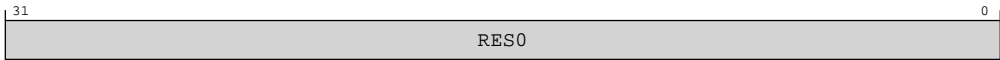


Table B-620: COREROM_DEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0xFAC	None

This interface is accessible as follows:

RO

B.2.2.5.11 COREROM_LAR, Core ROM table Software Lock Access Register

No software locking implemented for ROM tables.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFB0

Access type

RESERVEDW

Reset value
0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions
Figure B-322: EXT_COREROM_LAR bit assignments

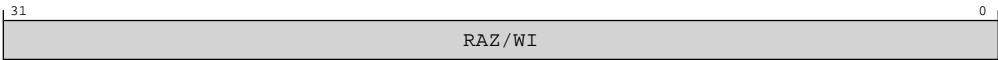


Table B-622: COREROM_LAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
COREROM	0xFB0	None

This interface is accessible as follows:
WO

B.2.2.5.12 COREROM_LSR, Core ROM table Software Lock Status Register

No software locking implemented for ROM tables.

Configurations
This register is available in all configurations.

Attributes

Width
32

Component
COREROM

Register offset
0xFB4

Access type
RO

Reset value
0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-323: EXT_COREROM_LSR bit assignments

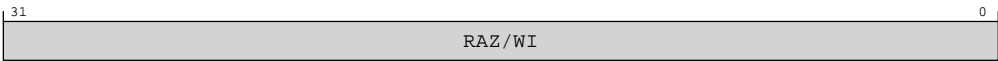


Table B-624: COREROM_LSR bit descriptions

Bits	Name	Description	Reset
[31:0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Range
COREROM	0xFB4	None

This interface is accessible as follows:

RO

B.2.2.5.13 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





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-324: EXT_COREROM_AUTHSTATUS bit assignments

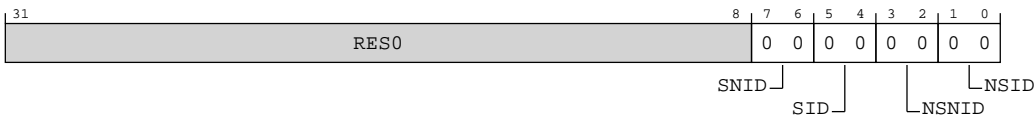


Table B-626: COREROM_AUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure Non-invasive Debug. 0b00 Debug level is not supported.	0b00
[5:4]	SID	Secure Invasive Debug. 0b00 Debug level is not supported.	0b00
[3:2]	NSNID	Non-secure Non-invasive Debug. 0b00 Debug level is not supported.	0b00
[1:0]	NSID	Non-secure Invasive Debug. 0b00 Debug level is not supported.	0b00

Accessibility

Component	Offset	Range
COREROM	0xFB8	None

This interface is accessible as follows:

RO

B.2.2.5.14 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-325: EXT_COREROM_DEVARCH bit assignments

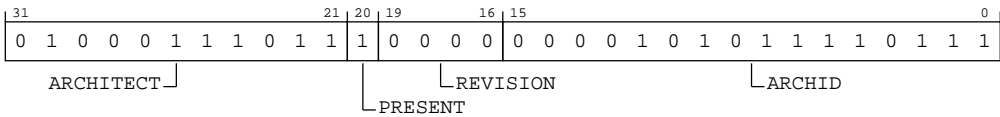


Table B-628: COREROM_DEVARCH bit descriptions

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.2.2.5.15 COREROM_DEVID2, Core ROM table Device Configuration Register 2

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFC0

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-326: EXT_COREROM_DEVID2 bit assignments

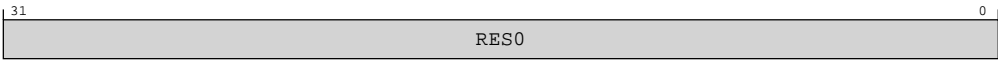


Table B-630: COREROM_DEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0xFC0	None

This interface is accessible as follows:

RO

B.2.2.5.16 COREROM_DEVID1, Core ROM table Device Configuration Register 1

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFC4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-327: EXT_COREROM_DEVID1 bit assignments

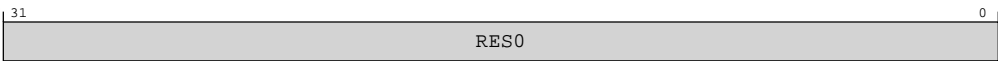


Table B-632: COREROM_DEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0xFC4	None

This interface is accessible as follows:

RO

B.2.2.5.17 COREROM_DEVID, Core ROM table Device Configuration Register

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFC8

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx00	0000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-328: EXT_COREROM_DEVID bit assignments

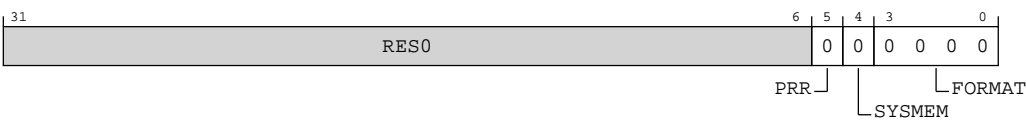


Table B-634: COREROM_DEVID bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	PRR	Power Request functionality included. 0b0 Power Request functionality not included.	0b0
[4]	SYSMEM	System memory present. 0b0 System memory is not present on the bus.	0b0
[3:0]	FORMAT	ROM format. 0b0000 32-bit format 0.	0b0000

Accessibility

Component	Offset	Range
COREROM	0xFC8	None

This interface is accessible as follows:

RO

B.2.2.5.18 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-329: EXT_COREROM_DEVTYPE bit assignments



Table B-636: COREROM_DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SUB	Sub number 0b0000 Other, undefined.	0b0000
[3:0]	MAJOR	Major number 0b0000 Miscellaneous.	0b0000

Accessibility

Component	Offset	Range
COREROM	0xFCC	None

This interface is accessible as follows:

RO

B.2.2.5.19 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-330: EXT_COREROM_PIDR4 bit assignments

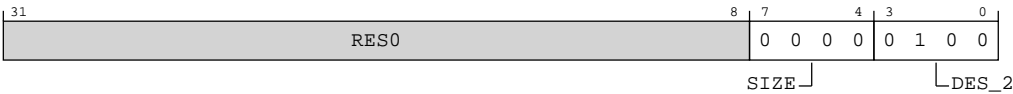


Table B-638: COREROM_PIDR4 bit descriptions

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.2.2.5.20 COREROM_PIDR5, Core ROM table Peripheral Identification Register 5

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFD4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-331: EXT_COREROM_PIDR5 bit assignments

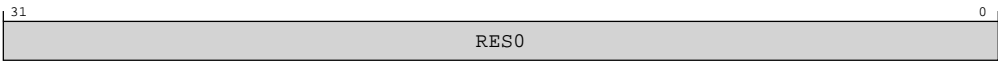


Table B-640: COREROM_PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0xFD4	None

This interface is accessible as follows:

RO

B.2.2.5.21 COREROM_PIDR6, Core ROM table Peripheral Identification Register 6

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

Register offset

0xFD8

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-332: EXT_COREROM_PIDR6 bit assignments

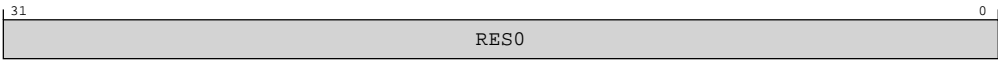


Table B-642: COREROM_PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0xFD8	None

This interface is accessible as follows:

RO

B.2.2.5.22 COREROM_PIDR7, Core ROM table Peripheral Identification Register 7

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

COREROM

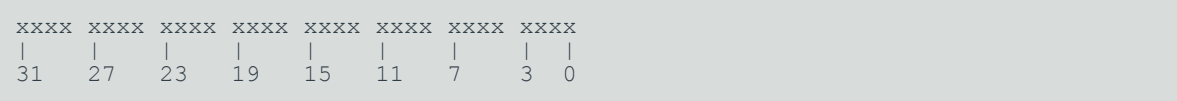
Register offset

0xFDC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-333: EXT_COREROM_PIDR7 bit assignments



Table B-644: COREROM_PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
COREROM	0xFDC	None

This interface is accessible as follows:

RO

B.2.2.5.23 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	0001	0101
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-334: EXT_COREROM_PIDR0 bit assignments

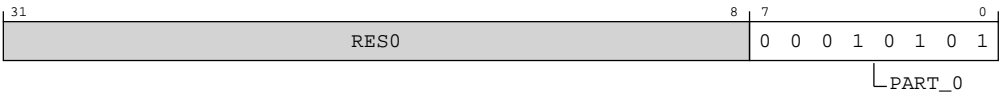


Table B-646: COREROM_PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b00010101 Cortex-R82 Core ROM table. Bits [7:0] of part number 0xD15.	0x15

Accessibility

Component	Offset	Range
COREROM	0xFE0	None

This interface is accessible as follows:

RO

B.2.2.5.24 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-335: EXT_COREROM_PIDR1 bit assignments

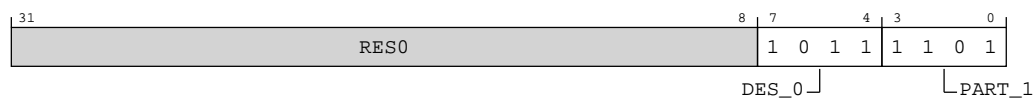


Table B-648: COREROM_PIDR1 bit descriptions

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 Cortex-R82 Core ROM table. Bits [11:8] of part number 0xD15.	0b1101

Accessibility

Component	Offset	Range
COREROM	0xFE4	None

This interface is accessible as follows:

RO

B.2.2.5.25 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

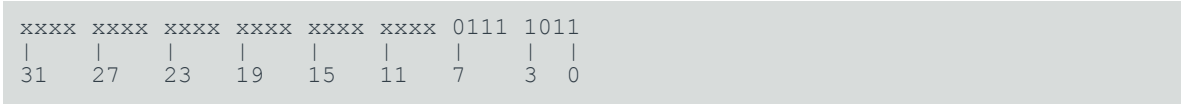
Register offset

0xFE8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-336: EXT_COREROM_PIDR2 bit assignments

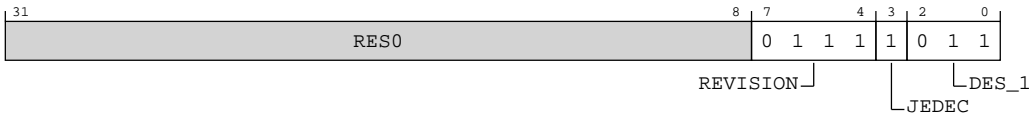


Table B-650: COREROM_PIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 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.2.2.5.26 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-337: EXT_COREROM_PIDR3 bit assignments

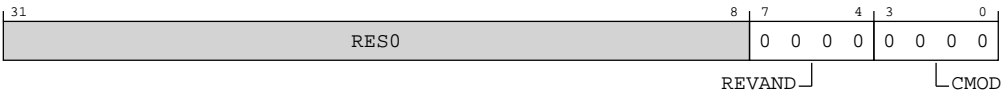


Table B-652: COREROM_PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Minor errata fixes. 0b0000 No ECO fixes.	0b0000
[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.2.2.5.27 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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-338: EXT_COREROM_CIDR0 bit assignments

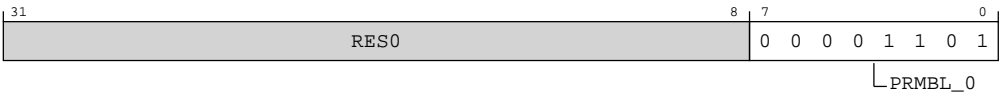


Table B-654: COREROM_CIDR0 bit descriptions

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.2.2.5.28 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-339: EXT_COREROM_CIDR1 bit assignments

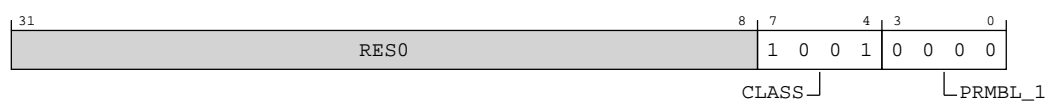


Table B-656: COREROM_CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	CoreSight component class. 0b1001 CoreSight component.	0b1001
[3:0]	PRMBL_1	CoreSight component identification preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
COREROM	0xFF4	None

This interface is accessible as follows:

RO

B.2.2.5.29 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

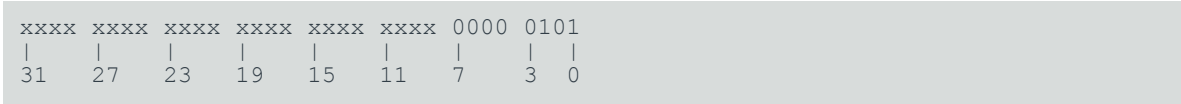
Register offset

0xFF8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-340: EXT_COREROM_CIDR2 bit assignments



Table B-658: COREROM_CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	CoreSight component identification preamble. 0b000000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
COREROM	0xFF8	None

This interface is accessible as follows:

RO

B.2.2.5.30 COREROM_CIDR3, Core ROM table Component Identification Register 3

Provides CoreSight discovery information.

Configurations

This register is available in all configurations.

Attributes

Width

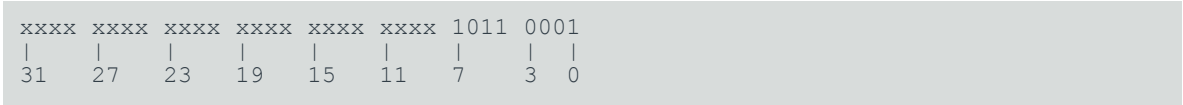
32

Component
COREROM

Register offset
0xFFC

Access type
RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-341: EXT_COREROM_CIDR3 bit assignments



Table B-660: 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.2.2.6 External CTI register description

This section includes the register descriptions for all memory-mapped *Cross Trigger Interface* (CTI) registers that are accessed for both the cluster and each core.

B.2.2.6.1 CTICONTROL, CTI Control register

Controls whether the CTI is enabled.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

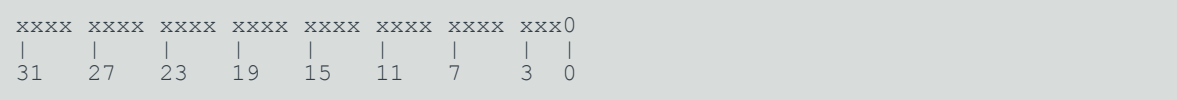
Register offset

0x000

Access type

RW

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-342: EXT_CTICONTROL bit assignments

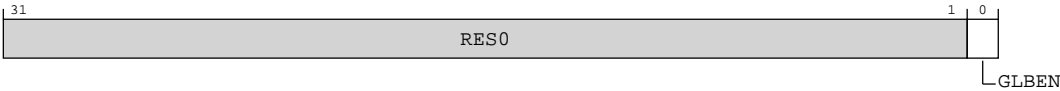


Table B-662: CTICONTROL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	GLBEN	<div>Enables or disables the CTI mapping functions. Possible values of this field are:</div> <div><div><div>0b0</div><div>CTI mapping functions and application trigger disabled.</div></div><div><div>0b1</div><div>CTI mapping functions and application trigger enabled.</div></div></div> <div>When GLBEN is 0, the input channel to output trigger, input trigger to output channel, and application trigger functions are disabled and do not signal new events on either output triggers or output channels. If a previously asserted output trigger has not been acknowledged, it is CONSTRAINED UNPREDICTABLE, and the implemented behavior is:</div> <div><div><div><div></div><div>The output trigger remains asserted after the mapping functions are disabled.</div></div></div></div> <div>All output triggers are disabled by CTI reset.</div> <div>If the ECT supports multicycle channel events any existing output channel events will be terminated.</div>	0b0

Accessibility

Component	Offset	Instance	Range
CTI	0x000	CTICONTROL	None

This interface is accessible as follows:

RW

B.2.2.6.2 CTIINTACK, CTI Output Trigger Acknowledge register

Can be used to deactivate the output triggers.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x010

Access type

Read

RESERVED

Write

W

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-343: EXT_CTIINTACK bit assignments

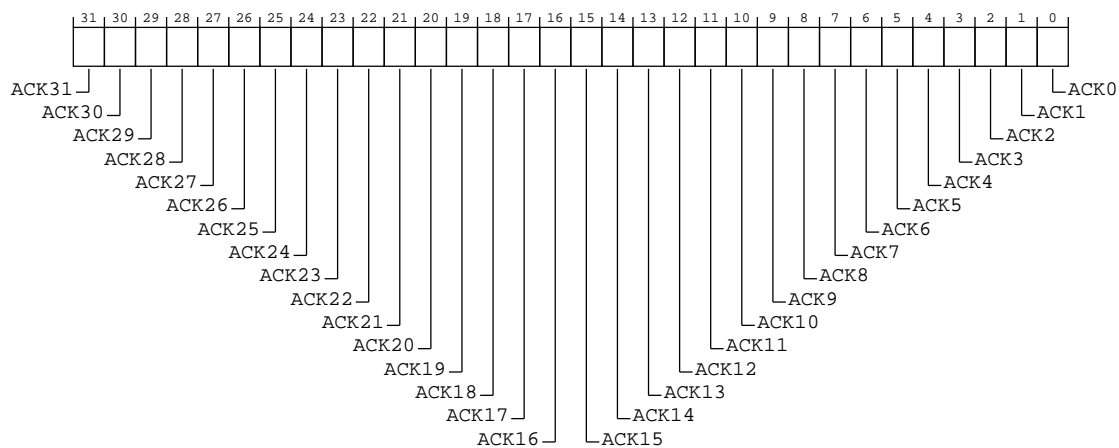


Table B-664: CTIINTACK bit descriptions

Bits	Name	Description	Reset
[31:0]	ACK<n>, bit[n], where n = 31 to 0	<p>Acknowledge for output trigger <n>.</p> <p>Bits [31:N] are RAZ/WI. N is the number of CTI triggers implemented as defined by the ext-CTIDEVID.NUMTRIG field.</p> <p>If any of the following is true, writes to ACK<n> are ignored:</p> <ul style="list-style-type: none"> n >= ext-CTIDEVID.NUMTRIG, the number of implemented triggers. Output trigger n is not active. The channel mapping function output, as controlled by ext-CTIOUTEN<n>, is still active. Output trigger n is not implemented. Output trigger n is not connected. Output trigger n is self-acknowledging and does not require software acknowledge. <p>Otherwise, the behavior on writes to ACK<n> is as follows:</p> <p>0b0 No effect</p> <p>0b1 Deactivate the trigger.</p>	32 {x}

Accessibility

A debugger must read ext-CTITRIGOUTSTATUS to confirm that the output trigger has been acknowledged before generating any event that must be ordered after the write to CTIINTACK, such as a write to CTIAPPPULSE to activate another trigger.

Component	Offset	Instance	Range
CTI	0x010	CTIINTACK	None

This interface is accessible as follows:

WO

B.2.2.6.3 CTIAPPSET, CTI Application Trigger Set register

Sets the application triggers.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x014

Access type

RW1S

Reset value

0000	0000	0000	0000	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-344: EXT_CTIAPPSET bit assignments

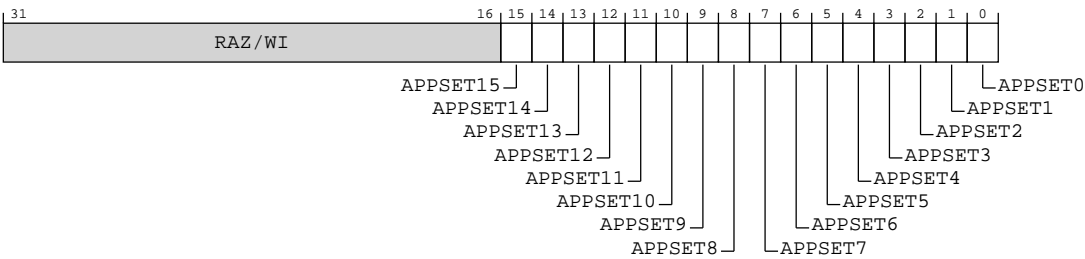


Table B-666: CTIAPPSET bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ/WI	Reserved	RAZ/WI
[15:0]	APPSET<x>, bit[x], where x = 15 to 0	<p>Application trigger <x> enable.</p> <p>Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field.</p> <p>0b0</p> <p>Reading this means the application trigger is inactive. Writing this has no effect.</p> <p>0b1</p> <p>Reading this means the application trigger is active. Writing this sets the corresponding application trigger to 1 and generates a channel event.</p> <p>If the ECT does not support multicycle channel events, use of CTIAPPSET is deprecated and the debugger must only use ext-CTIAPPULSE.</p>	16{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x014	CTIAPPSET	None

This interface is accessible as follows:

RW

B.2.2.6.4 CTIAPPCLEAR, CTI Application Trigger Clear register

Clears the application triggers.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x018

Access type

RESERVEDW

Reset value

0000	0000	0000	0000	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-345: EXT_CTIAPPCLEAR bit assignments

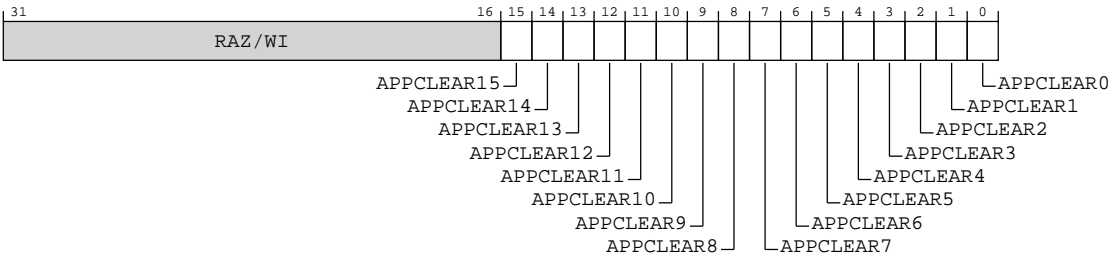


Table B-668: CTIAPPCLEAR bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ/WI	Reserved	RAZ/WI
[15:0]	APPCLEAR<x>, bit[x], where x = 15 to 0	<p>Application trigger <x> disable.</p> <p>Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field.</p> <p>Writing to this bit has the following effect:</p> <p>0b0</p> <p>No effect.</p> <p>0b1</p> <p>Clear corresponding application trigger to 0 and clear the corresponding channel event.</p> <p>If the ECT does not support multicycle channel events, use of CTIAPPCLEAR is deprecated and the debugger must only use ext-CTIAPPPULSE.</p>	16{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x018	CTIAPPCLEAR	None

This interface is accessible as follows:

WO

B.2.2.6.5 CTIAPPPULSE, CTI Application Pulse register

Causes event pulses to be generated on ECT channels.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x01C

Access type

Read

RESERVED

Write

W

Reset value

0000	0000	0000	0000	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-346: EXT_CTIAPPPULSE bit assignments

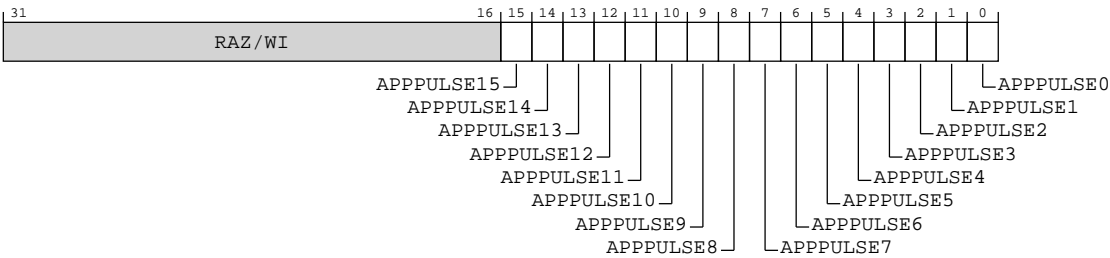


Table B-670: CTIAPPPULSE bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ/WI	Reserved	RAZ/WI

Bits	Name	Description	Reset
[15:0]	APPPULSE<x>, bit[x], where x = 15 to 0	<p>Generate event pulse on ECT channel <x>.</p> <p>Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field.</p> <p>Writing to this bit has the following effect:</p> <p>0b0</p> <p>No effect.</p> <p>0b1</p> <p>Channel <x> event pulse generated.</p> <ul style="list-style-type: none"> The CTIAPPPULSE operation does not affect the state of the application trigger. If the channel is active, either because of an earlier event or from the application trigger, then the value written to CTIAPPPULSE might have no effect. Multiple pulse events that occur close together might be merged into a single pulse event. 	16 {x}

Accessibility

It is CONSTRAINED UNPREDICTABLE whether a write to CTIAPPPULSE generates an event on a channel if CTICONTROL.GLBEN is 0.

Component	Offset	Instance	Range
CTI	0x01C	CTIAPPPULSE	None

This interface is accessible as follows:

WO

B.2.2.6.6 CTIINEN<n>, CTI Input Trigger to Output Channel Enable registers, n = 0 - 9

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

Configurations

If input trigger n is not connected, the programming of CTIINEN<n> is ignored.

Attributes

Width

32

Component

CTI

Register offset

0x020 + (4 * n)

Access type

RW

Reset value

0000	0000	0000	0000	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-347: EXT_CTIINEN<n> bit assignments

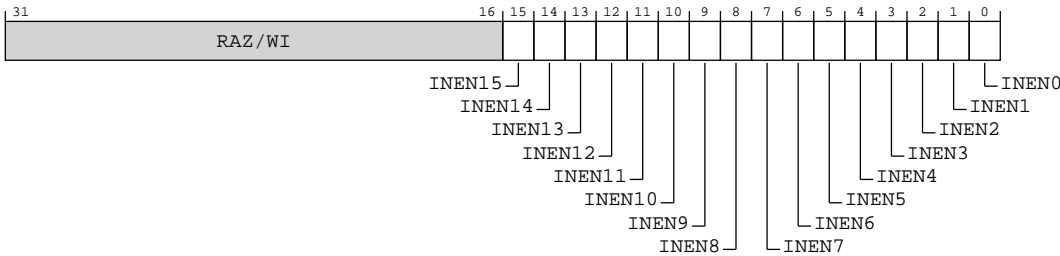


Table B-672: CTIINEN<n> bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ/WI	Reserved	RAZ/WI
[15:0]	INEN<x>, bit[x], where x = 15 to 0	Input trigger <n> to output channel <x> enable. Bits [31:N] are RAZ/WI . N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field. 0b0 Input trigger <n> will not generate an event on output channel <x>. 0b1 Input trigger <n> will generate an event on output channel <x>.	16 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0x020 + (4 * n)	CTIINEN<n>	None

This interface is accessible as follows:

RW

B.2.2.6.7 CTIOUTEN<n>, CTI Input Channel to Output Trigger Enable registers, n = 0 - 9

Defines which input channels generate output trigger n.

Configurations

If output trigger n is not connected, the programming of CTIOUTEN<n> is ignored.

Attributes

Width

32

Component

CTI

Register offset

0x0A0 + (4 * n)

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-348: EXT_CTIOUTEN<n> bit assignments

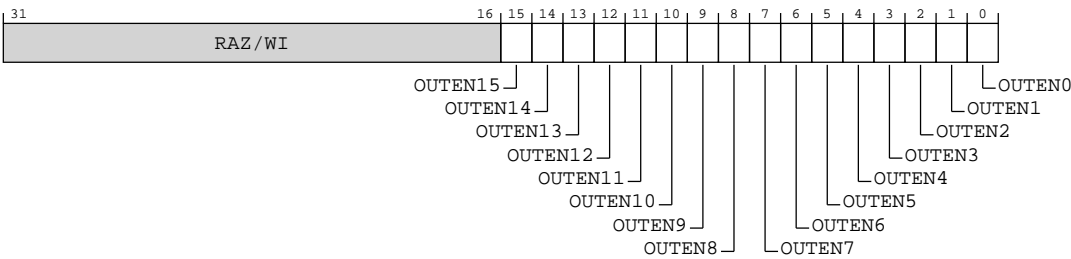


Table B-674: CTIOUTEN<n> bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ/WI	Reserved	RAZ/WI

Bits	Name	Description	Reset
[15:0]	OUTEN<x>, bit[x], where x = 15 to 0	<p>Input channel <x> to output trigger <n> enable.</p> <p>Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field.</p> <p>Possible values of this bit are:</p> <p>0b0</p> <p>An event on input channel <x> will not cause output trigger <n> to be asserted.</p> <p>0b1</p> <p>An event on input channel <x> will cause output trigger <n> to be asserted.</p>	16 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0x0A0 + (4 * n)	CTIOUTEN<n>	None

This interface is accessible as follows:

RW

B.2.2.6.8 CTITRIGINSTATUS, CTI Trigger In Status register

Provides the status of the trigger inputs.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x130

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
31	27	23	19	15	11	7	3 0

**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-349: EXT_CTITRIGINSTATUS bit assignments

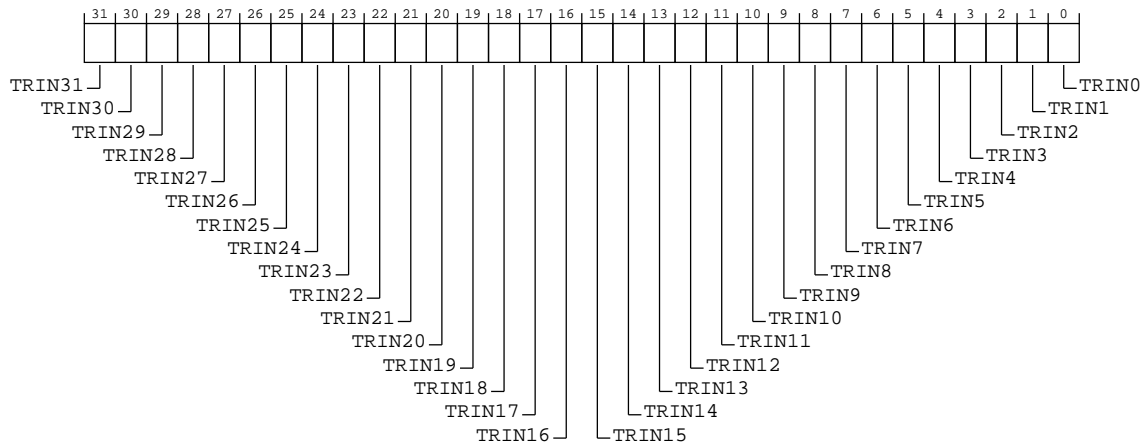


Table B-676: CTITRIGINSTATUS bit descriptions

Bits	Name	Description	Reset
[31:0]	TRIN<n>, bit[n], where n = 31 to 0	<p>Trigger input <n> status.</p> <p>Bits [31:N] are RAZ. N is the number of CTI triggers implemented as defined by the ext-CTIDEVID.NUMTRIG field.</p> <p>0b0</p> <p>Input trigger n is inactive.</p> <p>0b1</p> <p>Input trigger n is active.</p> <p>Not implemented and not-connected input triggers are always inactive.</p> <p>It is IMPLEMENTATION DEFINED whether an input trigger that does not support multicyle events can be observed as active.</p> <p>Input triggers that do not support multicyle events might be observed as active.</p>	32 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0x130	CTITRIGINSTATUS	None

This interface is accessible as follows:

RO

B.2.2.6.9 CTITRIGOUTSTATUS, CTI Trigger Out Status register

Provides the raw status of the trigger outputs, after processing by any **IMPLEMENTATION DEFINED** trigger interface logic. For output triggers that are self-acknowledging, this is only meaningful if the CTI implements multicycle channel events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x134

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-350: EXT_CTITRIGOUTSTATUS bit assignments

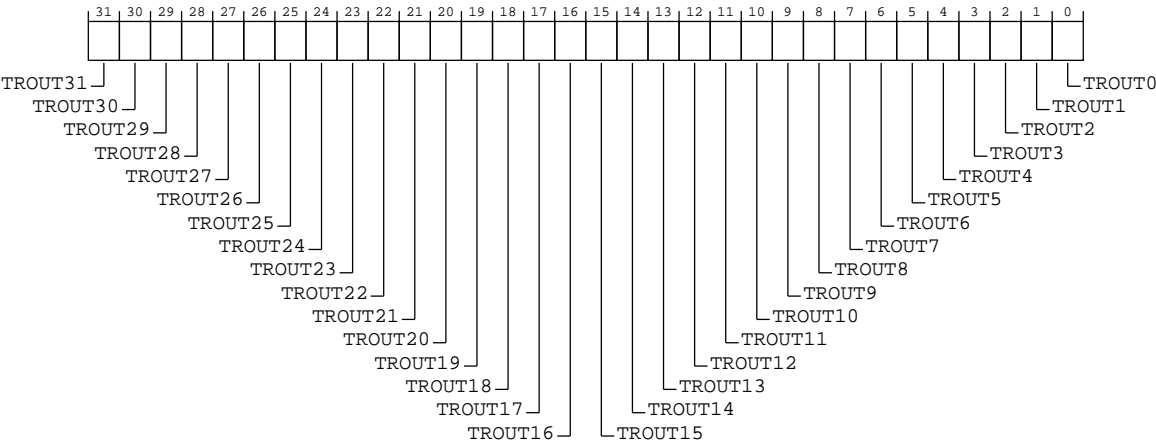


Table B-678: CTITRIGOUTSTATUS bit descriptions

Bits	Name	Description	Reset
[31:0]	TROUT<n>, bit[n], where n = 31 to 0	<p>Trigger output <n> status.</p> <p>Bits [31:N] are RAZ. N is the value in ext-CTIDEVID.NUMTRIG.</p> <p>If n < N, and output trigger <n> is implemented and connected, and either the trigger is not self-acknowledging or the CTI implements multicycle channel events, then permitted values for TROUT<n> are:</p> <p>0b0</p> <p>Output trigger n is inactive.</p> <p>0b1</p> <p>Output trigger n is active.</p> <p>Otherwise when n < N TROUT>n< is RAZ.</p>	32 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0x134	CTITRIGOUTSTATUS	None

This interface is accessible as follows:

RO

B.2.2.6.10 CTICHINSTATUS, CTI Channel In Status register

Provides the raw status of the ECT channel inputs to the CTI.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x138

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-351: EXT_CTICHINSTATUS bit assignments

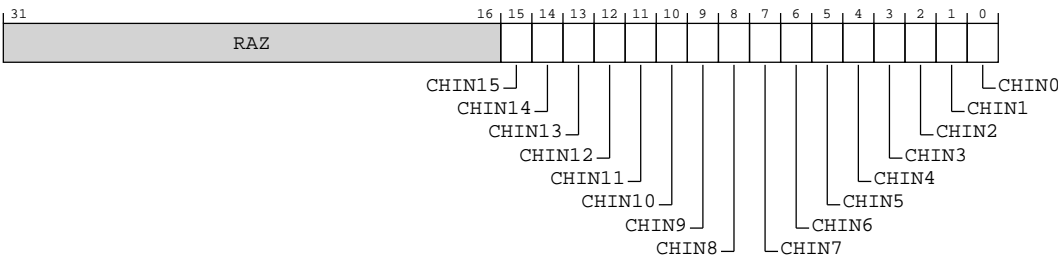


Table B-680: CTICHINSTATUS bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ	Reserved	RAZ

Bits	Name	Description	Reset
[15:0]	CHIN<n>, bit[n], where n = 15 to 0	<div>Input channel <n> status.</div> <div>Bits [31:N] are RAZ. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field.</div> <div>0b0 Input channel <n> is inactive.</div> <div>0b1 Input channel <n> is active.</div>	16 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0x138	CTICHINSTATUS	None

This interface is accessible as follows:

RO

B.2.2.6.11 CTICHOUTSTATUS, CTI Channel Out Status register

Provides the status of the ECT channel outputs from the CTI.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x13C

Access type

RO

Reset value

0000	0000	0000	0000	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-352: EXT_CTICHOUTSTATUS bit assignments

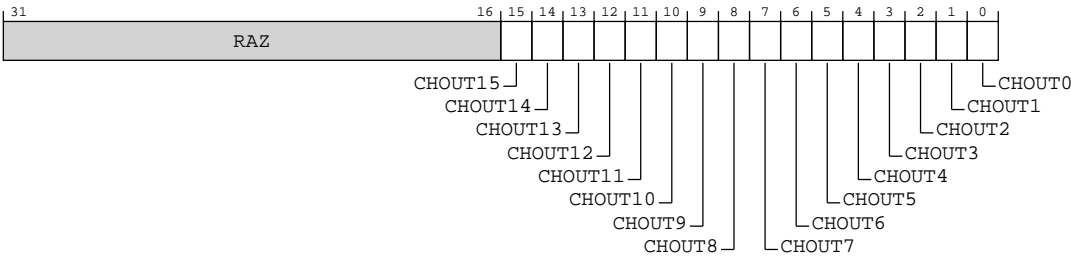


Table B-682: CTICHOUTSTATUS bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ	Reserved	RAZ
[15:0]	CHOUT<n>, bit[n], where n = 15 to 0	<p>Output channel <n> status.</p> <p>Bits [31:N] are RAZ. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field.</p> <p>Possible values of this bit are:</p> <p>0b0</p> <p>Output channel <n> is inactive.</p> <p>0b1</p> <p>Output channel <n> is active.</p> <p>Note:</p> <p>The value in CTICHOUTSTATUS is after gating by the channel gate. For more information, see ext-CTIGATE.</p>	16{x}

Accessibility

Component	Offset	Instance	Range
CTI	0x13C	CTICHOUTSTATUS	None

This interface is accessible as follows:

RO

B.2.2.6.12 CTIGATE, CTI Channel Gate Enable register

Determines whether events on channels propagate through the CTM to other ECT components, or from the CTM into the CTI.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0x140

Access type

RW

Reset value

0000	0000	0000	0000	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-353: EXT_CTIGATE bit assignments

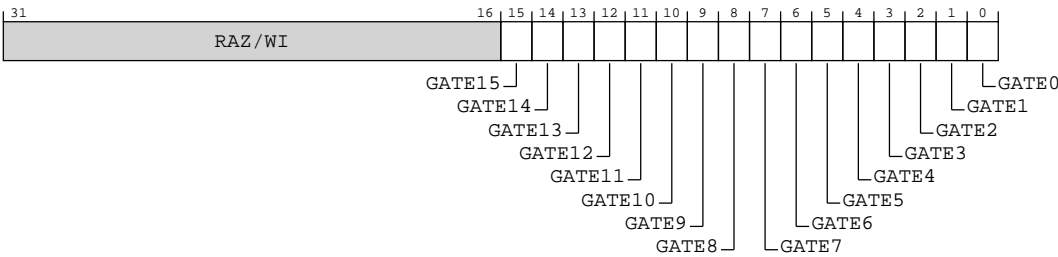


Table B-684: CTIGATE bit descriptions

Bits	Name	Description	Reset
[31:16]	RAZ/WI	Reserved	RAZ/WI

Bits	Name	Description	Reset
[15:0]	GATE<x>, bit[x], where x = 15 to 0	<p>Channel <x> gate enable.</p> <p>Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the ext-CTIDEVID.NUMCHAN field.</p> <p>0b0 Disable output and, if ext-CTIDEVID.INOUT == 0b01, input channel <x> propagation.</p> <p>0b1 Enable output and, if ext-CTIDEVID.INOUT == 0b01, input channel <x> propagation.</p> <p>If GATE<x> is set to 0, no new events will be propagated to the ECT, and if the ECT supports multicycle channel events any existing output channel events will be terminated.</p>	16 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0x140	CTIGATE	None

This interface is accessible as follows:

RW

B.2.2.6.13 ASICCTL, CTI External Multiplexer Control register

Can be used to provide **IMPLEMENTATION DEFINED** controls for the CTI. For example, the register might be used to control multiplexors for additional **IMPLEMENTATION DEFINED** triggers. The **IMPLEMENTATION DEFINED** controls provided by this register might modify the architecturally defined behavior of the CTI.



The architecturally-defined triggers must not be multiplexed.

Configurations

In this implementation, this register is in the Debug power domain

Attributes

Width

32

Component

CTI

Register offset

0x144

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-354: EXT_ASICCTL bit assignments



Table B-686: ASICCTL bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
CTI	0x144	ASICCTL	None

This interface is accessible as follows:

ImplementationDefined

B.2.2.6.14 CTIDEVCTL, CTI Device Control register

Provides target-specific device controls

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

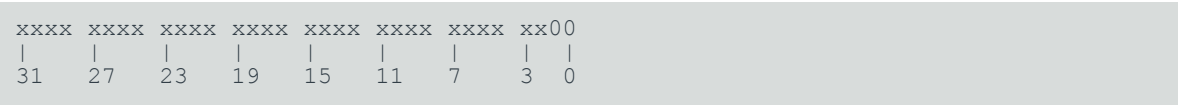
Register offset

0x150

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-355: EXT_CTIDEVCTL bit assignments

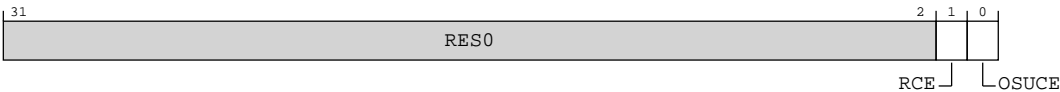


Table B-688: CTIDEVCTL bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	RCE	Reset Catch Enable. 0b0 Reset Catch debug event disabled. 0b1 Reset Catch debug event enabled.	0b0
[0]	OSUCE	OS Unlock Catch Enable 0b0 OS Unlock Catch debug event disabled. 0b1 OS Unlock Catch debug event enabled.	0b0

Accessibility

Component	Offset	Instance	Range
CTI	0x150	CTIDEVCTL	None

This interface is accessible as follows:

RW

B.2.2.6.15 CTIITCTRL, CTI Integration mode Control register

Enables the CTI to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

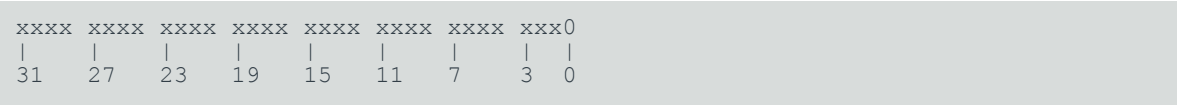
Register offset

0xF00

Access type

See bit descriptions

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-356: EXT_CTIITCTRL bit assignments

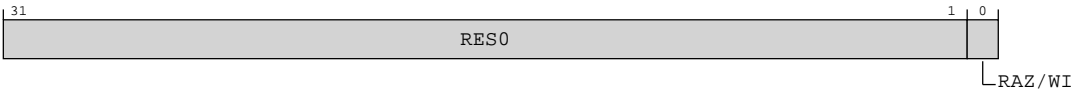


Table B-690: CTIITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Instance	Range
CTI	0xF00	CTIITCTRL	None

This interface is accessible as follows:

When IsCorePowered() and !OSLockStatus()

RW

Otherwise

ImplementationDefined

B.2.2.6.16 CTICLAIMSET, CTI Claim Tag Set register

Used by software to set CLAIM bits to 1.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFA0

Access type

RAOW1S

Reset value

0000 0000 0000 0000 0000 0000 0000 1111

Bit descriptions

Figure B-357: EXT_CTICLAIMSET bit assignments

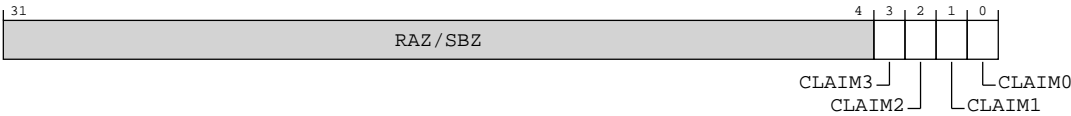


Table B-692: CTICLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:4]	RAZ/SBZ	Reserved	RAZ/SBZ

Bits	Name	Description	Reset
[3:0]	CLAIM<x>, bit[x], where x = 3 to 0	<div>CLAIM tag set bit.</div> <div>The bit is RAO and the behavior on writes is:</div> <div>0b0<div>No action.</div></div> <div>0b1<div>Indirectly set claim bit to 1.</div></div> <div>A single write to CTICLAIMSET can set multiple tags to 1.</div>	0b1111 ²⁹

Accessibility

Component	Offset	Instance	Range
CTI	0xFA0	CTICLAIMSET	None

This interface is accessible as follows:

RW

B.2.2.6.17 CTICLAIMCLR, CTI Claim Tag Clear register

Used by software to read the values of the CLAIM bits, and to clear these bits to 0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFA4

Access type

RW1C

Reset value

0000	0000	0000	0000	0000	0000	0000	xxxx
31	27	23	19	15	11	7	3 0

²⁹ An External Debug reset clears the CLAIM tag bits to 0.

Copyright © 2022–2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-358: EXT_CTICLAIMCLR bit assignments

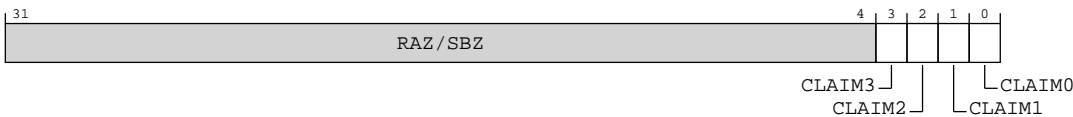


Table B-694: CTICLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:4]	RAZ/SBZ	Reserved	RAZ/SBZ
[3:0]	CLAIM<x>, bit[x], where x = 3 to 0	<p>CLAIM tag clear bit.</p> <p>Reads return the value of CLAIM[x] and the behavior on writes is:</p> <p>0b0</p> <p>No action.</p> <p>0b1</p> <p>Indirectly clear claim bit to 0.</p> <p>A single write to CTICLAIMCLR can clear multiple tags to 0.</p>	xxxx ³⁰

Accessibility

Component	Offset	Instance	Range
CTI	0xFA4	CTICLAIMCLR	None

This interface is accessible as follows:

RW

B.2.2.6.18 CTIDEVAFF0, CTI Device Affinity register 0

Copy of the low half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

Configurations

This register is available in all configurations.

³⁰ An External Debug reset clears the CLAIM tag bits to 0.

Copyright © 2022–2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Attributes

Width

32

Component

CTI

Register offset

0xFA8

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-359: EXT_CTIDEVAFF0 bit assignments

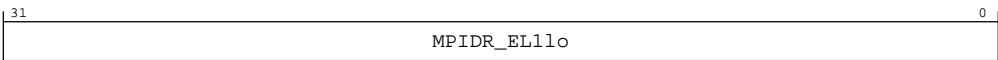


Table B-696: CTIDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1lo	If the CTI corresponds to a PE, then this field is a read-only copy of the low half of the core's AArch64-MPIDR_EL1, as seen from the highest implemented Exception level. If the CTI corresponds to the cluster, then this field is a read-only copy of the low half of any of the cores' AArch64-MPIDR_EL1, as seen from the highest implemented Exception level, but with bits [15:8] set to 0x80. In other words, the Aff1 field points to the cluster level, instead of a specific core in the cluster.	32 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0xFA8	CTIDEVAFF0	None

This interface is accessible as follows:

RO

B.2.2.6.19 CTIDEVAFF1, CTI Device Affinity register 1

Copy of the high half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFAC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-360: EXT_CTIDEVAFF1 bit assignments

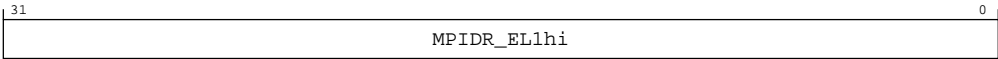


Table B-698: CTIDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1hi	<p>If the CTI corresponds to a PE, then this field is a read-only copy of the high half of the core's AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.</p> <p>If the CTI corresponds to the cluster, then this field is a read-only copy of the high half of any of the cores' AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.</p>	32 {x}

Accessibility

Component	Offset	Instance	Range
CTI	0xFAC	CTIDEVAFF1	None

This interface is accessible as follows:

RO

B.2.2.6.20 CTILAR, CTI Lock Access Register

Allows or disallows access to the CTI registers through a memory-mapped interface.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Cross-Trigger Interface registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Cross-Trigger Interface registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

Software Lock is not implemented.

Attributes

Width

32

Component

CTI

Register offset

0xFB0

Access type

RESERVEDW

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

Otherwise

Figure B-361: EXT_CTILAR bit assignments

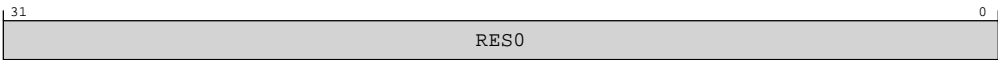


Table B-700: CTILAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
CTI	0xFB0	CTILAR	None

This interface is accessible as follows:

WO

B.2.2.6.21 CTILSR, CTI Lock Status Register

Indicates the current status of the Software Lock for CTI registers.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Cross-Trigger Interface registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Cross-Trigger Interface registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

Software Lock is not implemented.

Attributes

Width

32

Component

CTI

Register offset

0xFB4

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-362: EXT_CTILSR bit assignments

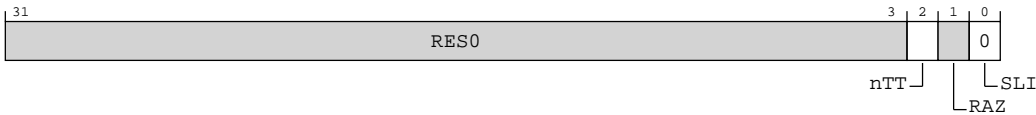


Table B-702: CTILSR bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0
[2]	nTT	Not thirty-two bit access required. RAZ .	x
[1]	RAZ	Reserved	RAZ
[0]	SLI	Software Lock implemented. 0b0 Software Lock not implemented or not memory-mapped access.	0b0

Accessibility

Component	Offset	Instance	Range
CTI	0xFB4	CTILSR	None

This interface is accessible as follows:

RO

B.2.2.6.22 CTIAUTHSTATUS, CTI Authentication Status register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for CTI.

Configurations

This register is **OPTIONAL**, and is required for CoreSight compliance.

Attributes

Width

32

Component

CTI

Register offset

0xFB8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-363: EXT_CTIAUTHSTATUS bit assignments

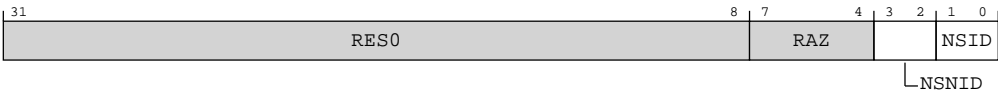


Table B-704: CTIAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	RAZ	Reserved	RAZ
[3:2]	NSNID	This field holds the same value as ext-DBGAUTHSTATUS_EL1.SNID.	xx
[1:0]	NSID	This field holds the same value as ext-DBGAUTHSTATUS_EL1.SID.	xx

Accessibility

Component	Offset	Instance	Range
CTI	0xFB8	CTIAUTHSTATUS	None

This interface is accessible as follows:

RO

B.2.2.6.23 CTIDEVARCH, CTI Device Architecture register

Identifies the programmers' model architecture of the CTI component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 0001 0001 1010 0001 0100

Bit descriptions

Figure B-364: EXT_CTIDEVARCH bit assignments

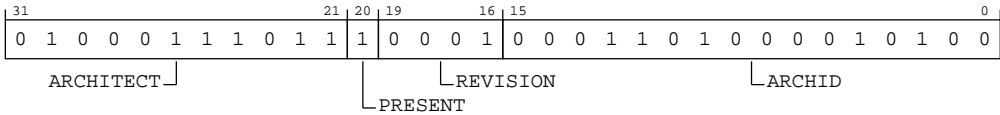


Table B-706: CTIDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For CTI, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B. 0b01000111011	0b01000111011
[20]	PRESENT	Indicates that the DEVARCH is present. 0b1	0b1
[19:16]	REVISION	Revision. Defines the architecture revision of the component. 0b0001 First revision, and also adds support for ext-CTIDEVCTL. All other values are reserved.	0b0001

Bits	Name	Description	Reset
[15:0]	ARCHID	<div>Defines this part to be an Armv8 debug component. For architectures defined by Arm this is further subdivided.</div> <div>For CTI:</div> <ul style="list-style-type: none">Bits [15:12] are the architecture version, 0x1.Bits [11:0] are the architecture part number, 0xA14. <div>This corresponds to CTI architecture version CTIv2.</div> <div>0b0001101000010100</div>	0x1A14

Accessibility

Component	Offset	Instance	Range
CTI	0xFBC	CTIDEVARCH	None

This interface is accessible as follows:

RO

B.2.2.6.24 CTIDEVID2, CTI Device ID register 2

Reserved for future information about the CTI component to the debugger.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFC0

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-365: EXT_CTIDEVID2 bit assignments

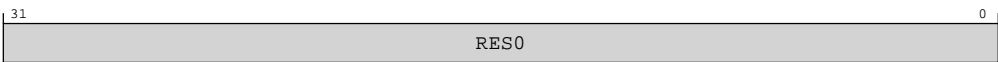


Table B-708: CTIDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
CTI	0xFC0	CTIDEVID2	None

This interface is accessible as follows:

RO

B.2.2.6.25 CTIDEVID1, CTI Device ID register 1

Reserved for future information about the CTI component to the debugger.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

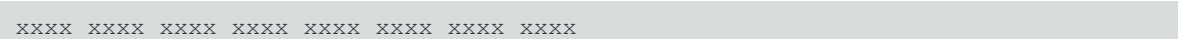
Register offset

0xFC4

Access type

RO

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-366: EXT_CTIDEVID1 bit assignments

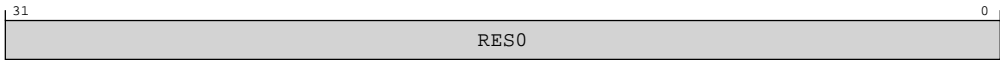


Table B-710: CTIDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
CTI	0xFC4	CTIDEVID1	None

This interface is accessible as follows:

RO

B.2.2.6.26 CTIDEVID, CTI Device ID register 0

Describes the CTI component to the debugger.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFC8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-367: EXT_CTIDEVID bit assignments

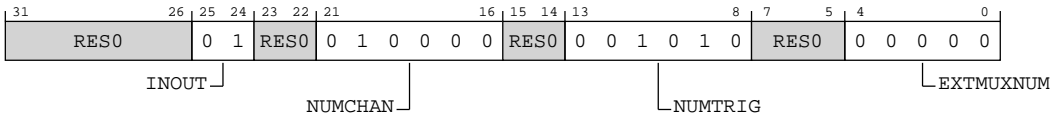


Table B-712: CTIDEVID bit descriptions

Bits	Name	Description	Reset
[31:26]	RES0	Reserved	RES0
[25:24]	INOUT	Input/output options. Indicates presence of the input gate. If the CTM is not implemented or CTIv2 is not implemented, this field is RAZ . 0b01 ext-CTIGATE masks propagation of input events from external channels. All other values are reserved.	0b01
[23:22]	RES0	Reserved	RES0
[21:16]	NUMCHAN	Number of ECT channels implemented. 0b010000 16 channels (0..15) implemented.	0b010000
[15:14]	RES0	Reserved	RES0
[13:8]	NUMTRIG	Number of triggers implemented. 0b001010 10 triggers (0..9) implemented.	0b001010
[7:5]	RES0	Reserved	RES0
[4:0]	EXTMUXNUM	Number of multiplexors available on triggers. This value is used in conjunction with External Control register, ext-ASICCTL. 0b00000 No multiplexors implemented. ext-ASICCTL is unused.	0b00000

Accessibility

Component	Offset	Instance	Range
CTI	0xFC8	CTIDEVID	None

This interface is accessible as follows:

RO

B.2.2.6.27 CTIDEVTYPE, CTI Device Type register

Indicates to a debugger that this component is part of a PE's cross-trigger interface.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

CTI

Register offset

0xFCC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-368: EXT_CTIDEVTYPE bit assignments

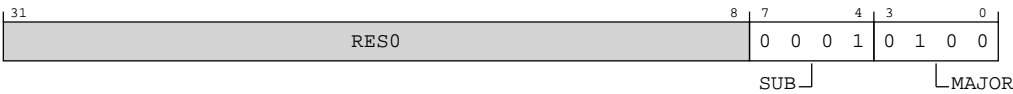


Table B-714: CTIDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	SUB	Subtype. Indicates this is a component within a PE. 0b0001	0b0001
[3:0]	MAJOR	Major type. Indicates this is a cross-trigger component. 0b0100	0b0100

Accessibility

Component	Offset	Instance	Range
CTI	0xFCC	CTIDEVTYPE	None

This interface is accessible as follows:

RO

B.2.2.6.28 CTIPIDR4, CTI Peripheral Identification Register 4

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

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-369: EXT_CTIPIDR4 bit assignments

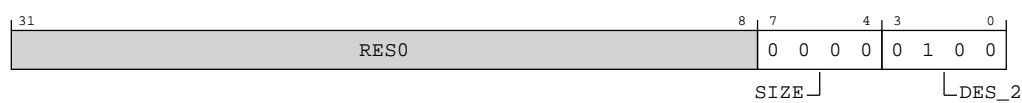


Table B-716: CTIPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	SIZE	Size of the component. RAZ . Log ₂ of the number of 4KB pages from the start of the component to the end of the component ID registers. 0b0000 The component uses a single 4KB block.	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. 0b0100 Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B.	0b0100

Accessibility

Component	Offset	Instance	Range
CTI	0xFD0	CTIPIDR4	None

This interface is accessible as follows:

RO

B.2.2.6.29 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

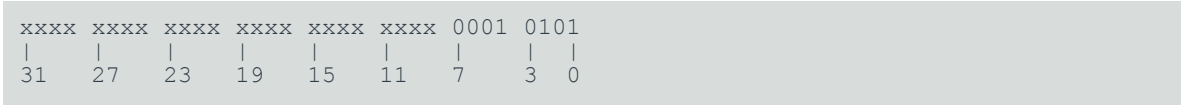
CTI

Register offset

0xFE0

Access type
RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-370: EXT_CTIPIDR0 bit assignments



Table B-718: CTIPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b00010101 Cortex-R82 CTI. Bits [7:0] of part number 0xD15.	0x15

Accessibility

Component	Offset	Instance	Range
CTI	0xFEO	CTIPIDR0	None

This interface is accessible as follows:

RO

B.2.2.6.30 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

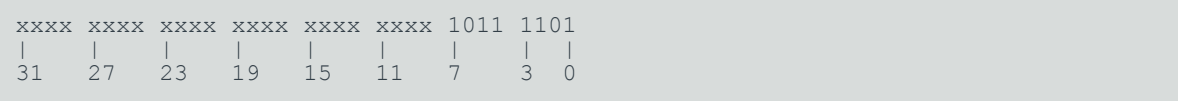
Register offset

0xFE4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-371: EXT_CTIPIDR1 bit assignments

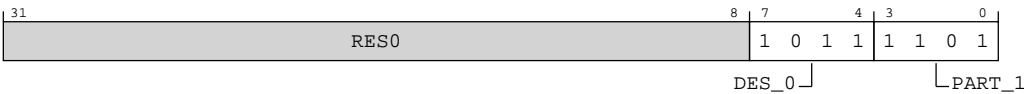


Table B-720: CTIPIDR1 bit descriptions

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 Arm Limited. Bits [3:0] of JEP106 identification code 0x3B.	0b1011
[3:0]	PART_1	Part number bits [11:8]. 0b1101 Cortex-R82 CTI. Bits [11:8] of part number 0xD15.	0b1101

Accessibility

Component	Offset	Instance	Range
CTI	0xFE4	CTIPIDR1	None

This interface is accessible as follows:

RO

B.2.2.6.31 CTIPIDR2, CTI Peripheral Identification Register 2

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

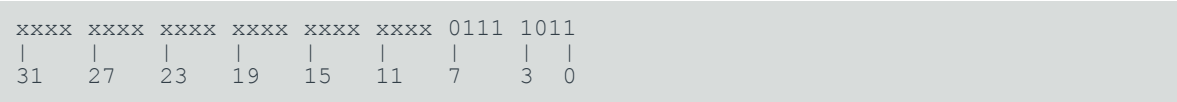
Register offset

0xFE8

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-372: EXT_CTIPIDR2 bit assignments

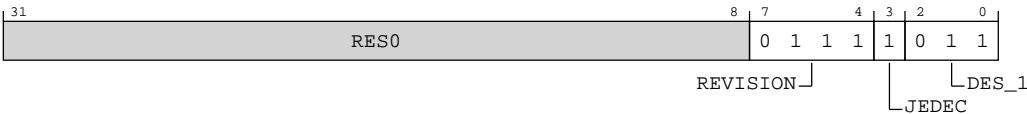


Table B-722: CTIPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111
[3]	JEDEC	RAO . Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Instance	Range
CTI	0xFE8	CTIPIDR2	None

This interface is accessible as follows:

RO

B.2.2.6.32 CTIPIDR3, CTI Peripheral Identification Register 3

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

0xFEC

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-373: EXT_CTIPIDR3 bit assignments



Table B-724: CTIPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Part minor revision. Parts using ext-CTIPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
CTI	0xFEC	CTIPIDR3	None

This interface is accessible as follows:

RO

B.2.2.6.33 CTICIDR0, CTI Component Identification Register 0

Provides information to identify a CTI 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

CTI

Register offset

0xFF0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-374: EXT_CTICIDR0 bit assignments

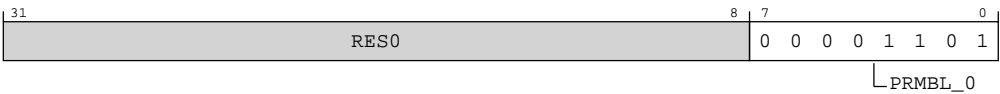


Table B-726: CTICIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble. 0b00001101 CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Instance	Range
CTI	0xFF0	CTICIDR0	None

This interface is accessible as follows:

RO

B.2.2.6.34 CTICIDR1, CTI Component Identification Register 1

Provides information to identify a CTI 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

CTI

Register offset

0xFF4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-375: EXT_CTICIDR1 bit assignments

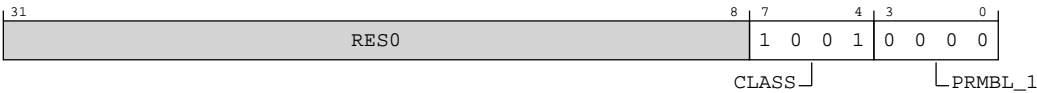


Table B-728: CTICIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component class. 0b1001 CoreSight debug component.	0b1001

Bits	Name	Description	Reset
[3:0]	PRMBL_1	Preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Instance	Range
CTI	0xFF4	CTICIDR1	None

This interface is accessible as follows:

RO

B.2.2.6.35 CTICIDR2, CTI Component Identification Register 2

Provides information to identify a CTI 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

CTI

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-376: EXT_CTICIDR2 bit assignments

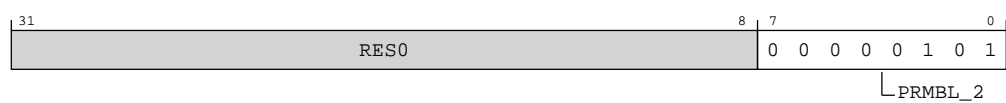


Table B-730: CTICIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble. 0b00000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Instance	Range
CTI	0xFF8	CTICIDR2	None

This interface is accessible as follows:

RO

B.2.2.6.36 CTICIDR3, CTI Component Identification Register 3

Provides information to identify a CTI 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

CTI

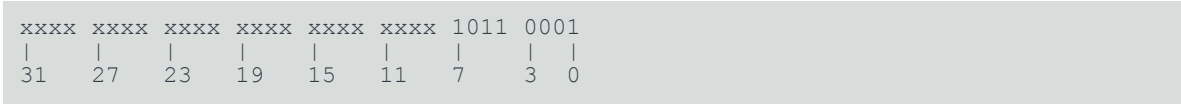
Register offset

0xFFC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-377: EXT_CTICIDR3 bit assignments



Table B-732: CTICIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble. 0b10110001 CoreSight component identification preamble.	0xB1

Accessibility

Component	Offset	Instance	Range
CTI	0xFFC	CTICIDR3	None

This interface is accessible as follows:

RO

B.2.2.7 External ETM register description

This section includes the register descriptions for all memory-mapped *Embedded Trace Macrocell* (ETM) registers that are accessed for each core.

B.2.2.7.1 TRCPRGCTLR, Programming Control Register

Enables the trace unit.

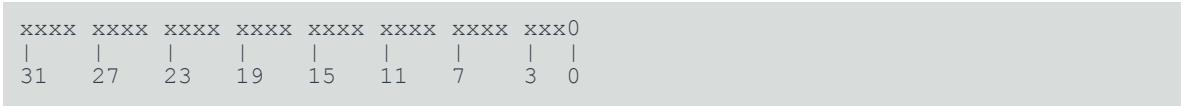
Configurations

External register TRCPRGCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.9 TRCPRGCTLR, Programming Control Register](#) on page 1236 bits [31:0].

Attributes

- Width
- 32
- Component
- ETM
- Register offset
- 0x004
- Access type
- Read
- R
- Write
- W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-378: EXT_TRCPRGCTLR bit assignments

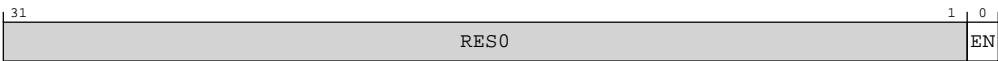


Table B-734: TRCPRGCTLR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	EN	Trace unit enable. 0b0 The trace unit is disabled. 0b1 The trace unit is enabled.	0b0

Accessibility

Must be programmed.

Component	Offset	Range
ETM	0x004	None

This interface is accessible as follows:

RW

B.2.2.7.2 TRCSTATR, Trace Status Register

Returns the trace unit status.

Configurations

External register TRCSTATR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.14 TRCSTATR, Trace Status Register](#) on page 1245 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x00C

Access type

RO

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-379: EXT_TRCSTATR bit assignments

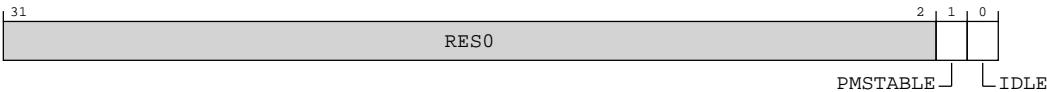


Table B-736: TRCSTATR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0
[1]	PMSTABLE	Programmers' model stable. 0b0 The programmers' model is not stable. 0b1 The programmers' model is stable. This bit is UNKNOWN while the trace unit is enabled.	x
[0]	IDLE	Idle status. 0b0 The trace unit is not idle. 0b1 The trace unit is idle.	x

Accessibility

Component	Offset	Range
ETM	0x00C	None

This interface is accessible as follows:

RO

B.2.2.7.3 TRCCONFIGR, Trace Configuration Register

Controls the tracing options.

Configurations

External register TRCCONFIGR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.16 TRCCONFIGR, Trace Configuration Register](#) on page 1248 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x010

Access type

Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-380: EXT_TRCCONFIGR bit assignments

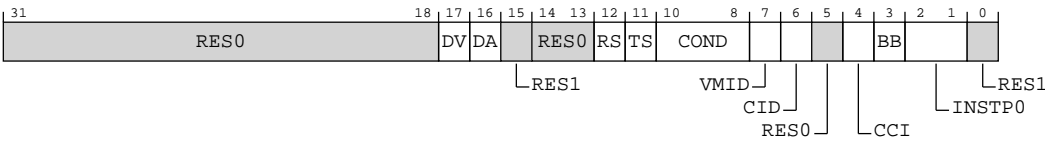


Table B-738: TRCCONFIGR bit descriptions

Bits	Name	Description	Reset
[31:18]	RES0	Reserved	RES0
[17]	DV	Data value tracing bit. 0b0 Data value tracing is disabled. 0b1 Data value tracing is enabled when INSTP0 is not 0b00.	x
[16]	DA	Data address tracing bit. 0b0 Data address tracing is disabled. 0b1 Data address tracing is enabled when INSTP0 is not 0b00.	x
[15]	RES1	Reserved	RES1

Bits	Name	Description	Reset
[14:13]	RES0	Reserved	RES0
[12]	RS	Return stack control. 0b0 Return stack is disabled. 0b1 Return stack is enabled.	x
[11]	TS	Global timestamp tracing control. 0b0 Global timestamp tracing is disabled. 0b1 Global timestamp tracing is enabled.	x
[10:8]	COND	Conditional instruction tracing bit. The permitted values are: 0b000 Conditional instruction tracing is disabled. 0b001 Conditional load instructions are traced. 0b010 Conditional store instructions are traced. 0b011 Conditional load and store instructions are traced. 0b111 All conditional instructions are traced.	xxx
[7]	VMID	Virtual context identifier tracing control. 0b0 Virtual context identifier tracing is disabled. 0b1 Virtual context identifier tracing is enabled.	x
[6]	CID	Context identifier tracing control. 0b0 Context identifier tracing is disabled. 0b1 Context identifier tracing is enabled.	x
[5]	RES0	Reserved	RES0
[4]	CCI	Cycle counting instruction tracing control. 0b0 Cycle counting instruction tracing is disabled. 0b1 Cycle counting instruction tracing is enabled.	x

Bits	Name	Description	Reset
[3]	BB	Branch broadcasting control. 0b0 Branch broadcasting is disabled. 0b1 Branch broadcasting is enabled.	x
[2:1]	INSTPO	Instruction PO field. This field controls whether load and store instructions are traced as PO instructions: 0b00 Do not trace load and store instructions as PO instructions. 0b01 Trace load instructions as PO instructions. 0b10 Trace store instructions as PO instructions. 0b11 Trace load and store instructions as PO instructions.	xx
[0]	RES1	Reserved	RES1

Access

Must always be programmed.

TRCCONFIGR.QE must be set to 0b00 if TRCCONFIGR.BB is not 0b0.

Accessibility

Must always be programmed.

TRCCONFIGR.QE must be set to 0b00 if TRCCONFIGR.BB is not 0b0.

Component	Offset	Range
ETM	0x010	None

This interface is accessible as follows:

RW

B.2.2.7.4 TRCAUXCTLR, Auxillary Control Register

Trace micro-architectural control bits.

Configurations

External register TRCAUXCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.19 TRCAUXCTLR, Auxillary Control Register](#) on page 1255 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x018

Access type

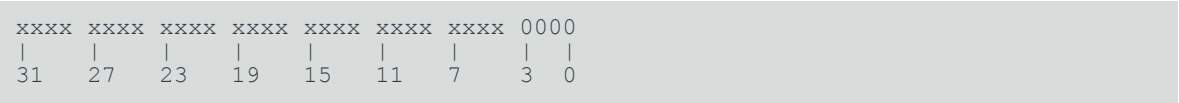
Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-381: EXT_TRCAUXCTLR bit assignments

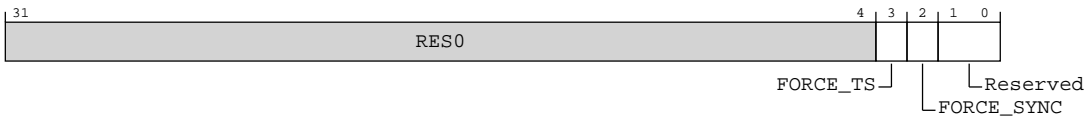


Table B-740: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3]	FORCE_TS	Force overflow when no timestamp is generated between two trace info packets. 0b0 Force overflow disabled. 0b1 Force overflow enabled.	0b0
[2]	FORCE_SYNC	Force overflow when the generation of trace synchronisation packets is delayed. 0b0 Force overflow disabled. 0b1 Force overflow enabled.	0b0

Bits	Name	Description	Reset
[1:0]	Reserved	Trace unit control options which are reserved for Arm internal use. 0b00 Any trace unit control options affected by this register are disabled. All other values are reserved for Arm internal use. Arm strongly recommends that you do not modify this field unless directed by Arm.	0b00

Accessibility

If this register is set to nonzero then it might cause the behavior of a trace unit to contradict the architecture specification. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

Component	Offset	Range
ETM	0x018	None

This interface is accessible as follows:

RW

B.2.2.7.5 TRCEVENTCTLOR, Event Control 0 Register

Controls the generation of tracing events.

Configurations

External register TRCEVENTCTLOR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.22 TRCEVENTCTLOR, Event Control 0 Register](#) on page 1262 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x020

Access type

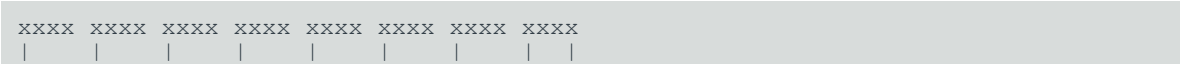
Read

R

Write

W

Reset value



31 27 23 19 15 11 7 3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-382: EXT_TRCEVENTCTL0R bit assignments

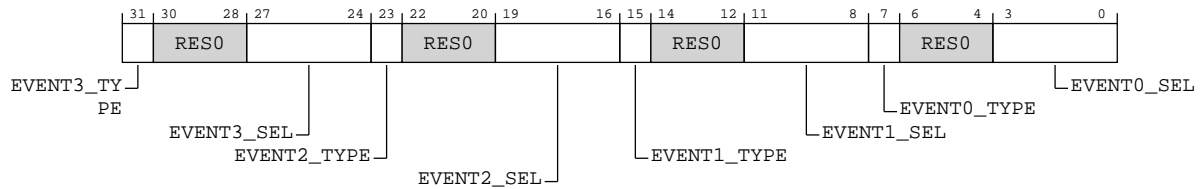


Table B-742: TRCEVENTCTL0R bit descriptions

Bits	Name	Description	Reset
[31]	EVENT3_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. EVENT3_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. EVENT3_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT3_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[30:28]	RES0	Reserved	RES0
[27:24]	EVENT3_SEL	Defines the selected Resource Selector or pair of Resource Selectors. EVENT3_TYPE controls whether EVENT3_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire. When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[3] == 0b1, then Event element 3 is generated in the instruction trace element stream.	xxxx

Bits	Name	Description	Reset
[23]	EVENT2_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT2_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT2_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT2_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[22:20]	RES0	Reserved	RES0
[19:16]	EVENT2_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT2_TYPE controls whether EVENT2_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p> <p>When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[2] == 0b1, then Event element 3 is generated in the instruction trace element stream.</p>	xxxx
[15]	EVENT1_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT1_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT1_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT1_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[14:12]	RES0	Reserved	RES0
[11:8]	EVENT1_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT1_TYPE controls whether EVENT1_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p> <p>When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[1] == 0b1, then Event element 3 is generated in the instruction trace element stream.</p>	xxxx

Bits	Name	Description	Reset
[7]	EVENTO_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. EVENTO_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. EVENTO_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENTO_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENTO_SEL	Defines the selected Resource Selector or pair of Resource Selectors. EVENTO_TYPE controls whether EVENTO_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire. When any of the selected resource events occurs and ext-TRCEVENTCTL1R.INSTEN[0] == 0b1, then Event element 3 is generated in the instruction trace element stream.	xxxx

Accessibility

Must be programmed if implemented.

Component	Offset	Range
ETM	0x020	None

This interface is accessible as follows:

RW

B.2.2.7.6 TRCEVENTCTL1R, Event Control 1 Register

Controls the behavior of the tracing events that ext-TRCEVENTCTL0R selects.

Configurations

External register TRCEVENTCTL1R bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.26 TRCEVENTCTL1R, Event Control 1 Register](#) on page 1273 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x024

Access type

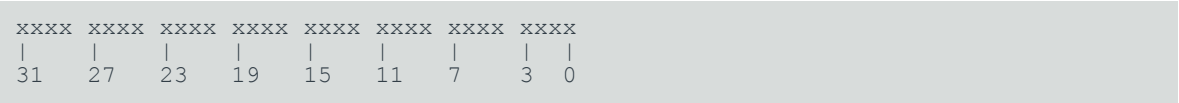
Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-383: EXT_TRCEVENTCTL1R bit assignments

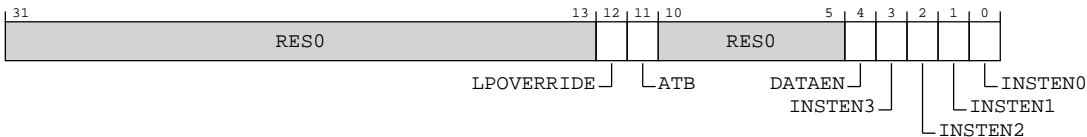


Table B-744: TRCEVENTCTL1R bit descriptions

Bits	Name	Description	Reset
[31:13]	RES0	Reserved	RES0
[12]	LPOVERRIDE	Low-power Override Mode select. 0b0 Trace unit Low-power Override Mode is not enabled. That is, the trace unit is permitted to enter low-power state. 0b1 Trace unit Low-power Override Mode is enabled. That is, entry to a low-power state does not affect the trace unit resources or trace generation.	x

Bits	Name	Description	Reset
[11]	ATB	<p>AMBA Trace Bus (ATB) trigger enable.</p> <p>When trace event 0 occurs the trace unit sets:</p> <ul style="list-style-type: none"> • ATID == 0x7D. • ATDATA to the value of ext-TRCTRACEIDR. <p>If the width of ATDATA is greater than the width of ext-TRCTRACEIDR.TRACEID then the trace unit zeros the upper ATDATA bits.</p> <p>If trace event 0 is programmed to occur based on program execution, such as an address comparator, the ATB trigger might not be inserted into the ATB stream at the same time as any trace generated by that program execution is output by the trace unit. Typically, the generated trace might be buffered in a trace unit which means that the ATB trigger would be output before the associated trace is output.</p> <p>If trace event 0 is asserted multiple times in close succession, the trace unit is required to generate an ATB trigger for the first assertion, but might ignore one or more of the subsequent assertions. Arm recommends that the window in which ETMEvent 0 is ignored is limited only by the time taken to output an ATB trigger.</p> <p>0b0 ATB trigger is disabled.</p> <p>0b1 ATB trigger is enabled.</p>	x
[10:5]	RES0	Reserved	RES0
[4]	DATAEN	<p>Data event enable bit.</p> <p>0b0 If event 0 occurs, the trace unit does not generate an Event element.</p> <p>0b1 If event 0 occurs, the trace unit generates an Event element in the data trace stream.</p>	x
[3:0]	INSTEN<m>, bit[m], where m = 3 to 0	<p>Event element control.</p> <p>0b0 The trace unit does not generate an Event element m.</p> <p>0b1 The trace unit generates an Event element m.</p>	xxxx

Accessibility

Must be programmed.

Component	Offset	Range
ETM	0x024	None

This interface is accessible as follows:

RW

B.2.2.7.7 TRCSTALLCTLR, Stall Control Register

Enables trace unit functionality that prevents trace unit buffer overflows.

Configurations

External register TRCSTALLCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.31 TRCSTALLCTLR, Stall Control Register](#) on page 1285 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x02C

Access type

Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-384: EXT_TRCSTALLCTLR bit assignments

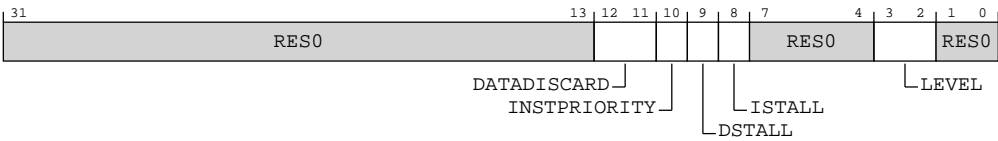


Table B-746: TRCSTALLCTLR bit descriptions

Bits	Name	Description	Reset
[31:13]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[12:11]	DATADISCARD	<p>Data discard field. Controls if a trace unit can discard data trace elements when the data trace buffer space is less than LEVEL.</p> <p>0b00 The trace unit must not discard any data trace elements.</p> <p>0b01 The trace unit can discard P1 and P2 elements associated with data loads.</p> <p>0b10 The trace unit can discard P1 and P2 elements associated with data stores.</p> <p>0b11 The trace unit can discard P1 and P2 elements associated with both data loads and stores.</p>	xx
[10]	INSTPRIORITY	<p>Prioritize instruction trace bit. Controls if a trace unit can prioritize instruction trace when the instruction trace buffer space is less than LEVEL.</p> <p>0b0 The trace unit must not prioritize instruction trace.</p> <p>0b1 The trace unit can prioritize instruction trace. A trace unit might prioritize</p>	x
[9]	DSTALL	<p>Data stall bit. Controls if a trace unit can stall the PE when the data trace buffer space is less than LEVEL.</p> <p>0b0 The trace unit must not stall the PE.</p> <p>0b1 The trace unit can stall the PE.</p>	x
[8]	ISTALL	<p>Instruction stall control. Controls if a trace unit can stall the PE when the trace buffer space is less than LEVEL.</p> <p>0b0 The trace unit must not stall the PE.</p> <p>0b1 The trace unit can stall the PE.</p>	x
[7:4]	RES0	Reserved	RES0
[3:2]	LEVEL	<p>Threshold level field. The field supports 4 monotonic levels from 0b00 to 0b11.</p> <p>0b00 Minimal invasion. This setting has a greater risk of a trace unit buffer overflow.</p> <p>0b01 Small invasion.</p> <p>0b10 Big invasion.</p> <p>0b11 Maximum invasion. Reduced risk of a trace unit buffer overflow.</p>	xx
[1:0]	RES0	Reserved	RES0

Accessibility

Must be programmed if implemented.

Component	Offset	Range
ETM	0x02C	None

This interface is accessible as follows:

RW

B.2.2.7.8 TRCTSCTLR, Timestamp Control Register

Controls the insertion of global timestamps in the trace stream.

Configurations

External register TRCTSCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.33 TRCTSCTLR, Timestamp Control Register](#) on page 1291 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x030

Access type

Read

R

Write

W

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-385: EXT_TRCTSCTLR bit assignments

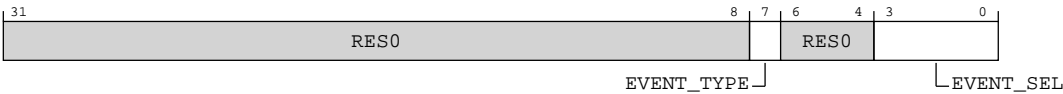


Table B-748: TRCTSCTLR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7]	EVENT_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. EVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. EVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. EVENT_TYPE controls whether EVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire.	xxxx

Accessibility

Must be programmed if ext-TRCCONFIGR.TS == 0b1.

Component	Offset	Range
ETM	0x030	None

This interface is accessible as follows:

RW

B.2.2.7.9 TRCSYNCP, Synchronization Period Register

Controls how often trace protocol synchronization requests occur.

Configurations

External register TRCSYNCP bits [31:0] are architecturally mapped to AArch64 System register A.2.8.35 TRCSYNCP, Synchronization Period Register on page 1295 bits [31:0].

Attributes

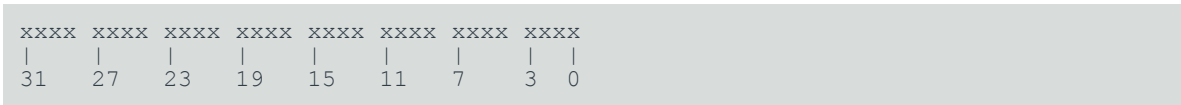
Width
32

Component
ETM

Register offset
0x034

Access type
Read
R
Write
W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-386: EXT_TRCSYNCP R bit assignments



Table B-750: TRCSYNCP R bit descriptions

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[4:0]	PERIOD	<p>Defines the number of bytes of trace between each periodic trace protocol synchronization request.</p> <p>0b000000 Trace protocol synchronization is disabled.</p> <p>0b010000 Trace protocol synchronization request occurs after 2^8 bytes of trace.</p> <p>0b01001 Trace protocol synchronization request occurs after 2^9 bytes of trace.</p> <p>0b01010 Trace protocol synchronization request occurs after 2^{10} bytes of trace.</p> <p>0b01011 Trace protocol synchronization request occurs after 2^{11} bytes of trace.</p> <p>0b01100 Trace protocol synchronization request occurs after 2^{12} bytes of trace.</p> <p>0b01101 Trace protocol synchronization request occurs after 2^{13} bytes of trace.</p> <p>0b01110 Trace protocol synchronization request occurs after 2^{14} bytes of trace.</p> <p>0b01111 Trace protocol synchronization request occurs after 2^{15} bytes of trace.</p> <p>0b10000 Trace protocol synchronization request occurs after 2^{16} bytes of trace.</p> <p>0b10001 Trace protocol synchronization request occurs after 2^{17} bytes of trace.</p> <p>0b10010 Trace protocol synchronization request occurs after 2^{18} bytes of trace.</p> <p>0b10011 Trace protocol synchronization request occurs after 2^{19} bytes of trace.</p> <p>0b10100 Trace protocol synchronization request occurs after 2^{20} bytes of trace.</p> <p>Other values are reserved. If a reserved value is programmed into PERIOD, then one of the following behaviors occurs:</p> <ul style="list-style-type: none"> If a reserved value less than 0b01000 is programmed, trace protocol synchronisation requests occur at approximately the specified period. If a reserved value greater than 0b10100 is programmed, no trace protocol synchronisation requests are generated. 	5 {x}

Accessibility

Must be programmed if ext-TRCIDR3.SYNCPR == 0b0.

Component	Offset	Range
ETM	0x034	None

This interface is accessible as follows:

RW

B.2.2.7.10 TRCCCCTLR, Cycle Count Control Register

Set the threshold value for cycle counting.

Configurations

External register TRCCCCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.37 TRCCCCTLR, Cycle Count Control Register](#) on page 1300 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x038

Access type

Read

R

Write

W

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-387: EXT_TRCCCCTLR bit assignments



Table B-752: TRCCCCTLR bit descriptions

Bits	Name	Description	Reset
[31:12]	RES0	Reserved	RES0
[11:0]	THRESHOLD	<p>Sets the threshold value for instruction trace cycle counting.</p> <p>The minimum threshold value that can be programmed into THRESHOLD is given in ext-TRCIDR3.CCITMIN. If the THRESHOLD value is smaller than the value in ext-TRCIDR3.CCITMIN, then no cycle counts are generated in the trace and the cycle count threshold is not included in a traceinfo packet.</p> <p>Writing a value of zero when ext-TRCCONFIGR.CCI is set to enable instruction trace cycle counting, then no cycle counts are generated in the trace and the cycle count threshold is not included in a traceinfo packet.</p>	12 {x}

Accessibility

Must be programmed if ext-TRCCONFIGR.CCI == 0b1.

Component	Offset	Range
ETM	0x038	None

This interface is accessible as follows:

RW

B.2.2.7.11 TRCBBCTLR, Branch Broadcast Control Register

Controls the regions in the memory map where branch broadcasting is active.

Configurations

External register TRCBBCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.39 TRCBBCTLR, Branch Broadcast Control Register](#) on page 1304 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x03C

Access type

Read

R

Write

W

Accessibility

Must be programmed if ext-TRCCONFIGR.BB == 0b1.

Component	Offset	Range
ETM	0x03C	None

This interface is accessible as follows:

RW

B.2.2.7.12 TRCTRACEIDR, Trace ID Register

Sets the trace ID for instruction trace.

Configurations

External register TRCTRACEIDR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.5 TRCTRACEIDR, Trace ID Register](#) on page 1226 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x040

Access type

Read

R

Write

W

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-389: EXT_TRCTRACEIDR bit assignments

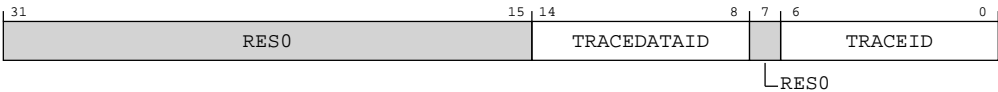


Table B-756: TRCTRACEIDR bit descriptions

Bits	Name	Description	Reset
[31:15]	RES0	Reserved	RES0
[14:8]	TRACEDATAID	<p>Data Trace ID.</p> <p>Sets the trace ID value for data trace.</p> <p>Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause UNPREDICTABLE behavior of the trace capture infrastructure.</p> <p>See the AMBA ATB Protocol Specification for information about which ATID values are reserved.</p>	7 {x}
[7]	RES0	Reserved	RES0
[6:0]	TRACEID	<p>Trace ID.</p> <p>Sets the trace ID value for instruction trace.</p> <p>Writing a reserved trace ID value does not affect behavior of the trace unit but it might cause UNPREDICTABLE behavior of the trace capture infrastructure.</p> <p>See the AMBA ATB Protocol Specification for information about which ATID values are reserved.</p>	7 {x}

Accessibility

Must be programmed if implemented.

Component	Offset	Range
ETM	0x040	None

This interface is accessible as follows:

RW

B.2.2.7.13 TRCVICTLR, ViewInst Main Control Register

Controls instruction trace filtering.

Configurations

External register TRCVICTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.6 TRCVICTLR, ViewInst Main Control Register](#) on page 1229 bits [31:0].

Attributes

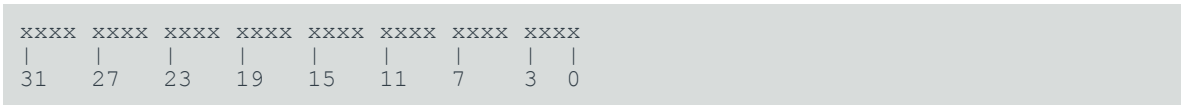
Width
32

Component
ETM

Register offset
0x080

Access type
Read
R
Write
W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-390: EXT_TRCVICTLR bit assignments

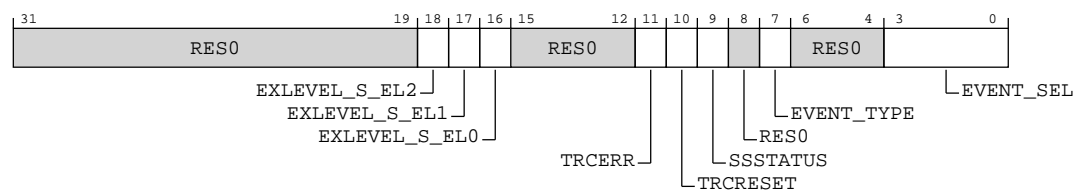


Table B-758: TRCVICTLR bit descriptions

Bits	Name	Description	Reset
[31:19]	RES0	Reserved	RES0
[18]	EXLEVEL_S_EL2	Filter instruction trace for EL2 in Secure state. 0b0 The trace unit generates instruction trace for EL2 in Secure state. 0b1 The trace unit does not generate instruction trace for EL2 in Secure state.	x

Bits	Name	Description	Reset
[17]	EXLEVEL_S_EL1	<p>Filter instruction trace for EL1 in Secure state.</p> <p>0b0</p> <p>The trace unit generates instruction trace for EL1 in Secure state.</p> <p>0b1</p> <p>The trace unit does not generate instruction trace for EL1 in Secure state.</p>	x
[16]	EXLEVEL_S_ELO	<p>Filter instruction trace for ELO in Secure state.</p> <p>0b0</p> <p>The trace unit generates instruction trace for ELO in Secure state.</p> <p>0b1</p> <p>The trace unit does not generate instruction trace for ELO in Secure state.</p>	x
[15:12]	RES0	Reserved	RES0
[11]	TRCERR	<p>Controls the forced tracing of System Error exceptions.</p> <p>0b0</p> <p>Forced tracing of System Error exceptions is disabled.</p> <p>0b1</p> <p>Forced tracing of System Error exceptions is enabled.</p>	x
[10]	TRCRESET	<p>Controls the forced tracing of PE Resets.</p> <p>0b0</p> <p>Forced tracing of PE Resets is disabled.</p> <p>0b1</p> <p>Forced tracing of PE Resets is enabled.</p>	x
[9]	SSSTATUS	<p>ViewInst start/stop function status.</p> <p>0b0</p> <p>Stopped State.</p> <p>The ViewInst start/stop function is in the stopped state.</p> <p>0b1</p> <p>Started State.</p> <p>The ViewInst start/stop function is in the started state.</p> <p>Before software enables the trace unit, it must write to this bit to set the initial state of the ViewInst start/stop function. If the ViewInst start/stop function is not used then set this bit to 0b1. Arm recommends that the value of this bit is set before each trace session begins.</p> <p>If the trace unit becomes disabled while a start point or stop point is still speculative, then the value of SSSTATUS is UNKNOWN and might represent the result of a speculative start point or stop point.</p> <p>If software which is running on the PE being traced disables the trace unit, either by clearing ext-TRCPRGCTLR.EN or locking the OS Lock, Arm recommends that a DSB and an ISB instruction are executed before disabling the trace unit to prevent any start points or stop points being speculative at the point of disabling the trace unit. This procedure assumes that all start points or stop points occur before the barrier instructions are executed. The procedure does not guarantee that there are no speculative start points or stop points when disabling, although it helps minimize the probability.</p>	x
[8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7]	EVENT_TYPE	<p>Chooses the type of Resource Selector.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>EVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>EVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENT_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. EVENT_TYPE controls whether EVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx

Accessibility

Must be programmed.

Component	Offset	Range
ETM	0x080	None

This interface is accessible as follows:

RW

B.2.2.7.14 TRCVIIECTLR, ViewInst Include/Exclude Control Register

Use this to select, or read, the Address Range Comparators for the ViewInst include/exclude function.

Configurations

External register TRCVIIECTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.10 TRCVIIECTLR, ViewInst Include/Exclude Control Register](#) on page 1238 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x084

Access type

Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-391: EXT_TRCVIIECTLR bit assignments

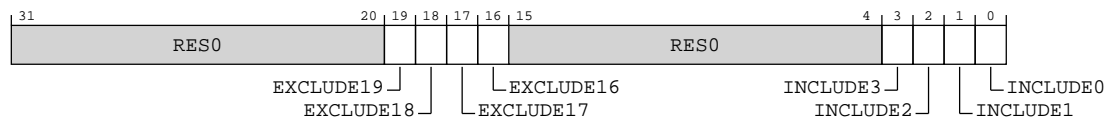


Table B-760: TRCVIIECTLR bit descriptions

Bits	Name	Description	Reset
[31:20]	RES0	Reserved	RES0
[19:16]	EXCLUDE<m>, bit[m], where m = 19 to 16	<p>Selects which Address Range Comparators are in use with the ViewInst exclude function.</p> <p>Each bit represents an Address Range Comparator, so bit[m] controls the selection of Address Range Comparator m.</p> <p>0b0</p> <p>The address range that Address Range Comparator m defines, is not selected for the ViewInst exclude function.</p> <p>0b1</p> <p>The address range that Address Range Comparator m defines, is selected for the ViewInst exclude function.</p>	xxxx
[15:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:0]	INCLUDE<m>, bit[m], where m = 3 to 0	<p>Selects which Address Range Comparators are in use with the ViewInst include function.</p> <p>Each bit represents an Address Range Comparator, so bit[m] controls the selection of Address Range Comparator m.</p> <p>Selecting no comparators for the ViewInst include function indicates that all instructions are included by default.</p> <p>The ViewInst exclude function then indicates which ranges are excluded.</p> <p>0b0</p> <p>The address range that Address Range Comparator m defines, is not selected for the ViewInst include function.</p> <p>0b1</p> <p>The address range that Address Range Comparator m defines, is selected for the ViewInst include function.</p>	xxxx

Accessibility

Must be programmed if ext-TRCIDR4.NUMACPAIRS > 0b0000.

Component	Offset	Range
ETM	0x084	None

This interface is accessible as follows:

RW

B.2.2.7.15 TRCVISSCTLR, ViewInst Start/Stop Control Register

Use this to select, or read, the Single Address Comparators for the ViewInst start/stop function.

Configurations

External register TRCVISSCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.12 TRCVISSCTLR, ViewInst Start/Stop Control Register](#) on page 1241 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x088

Access type

Read

R

Write

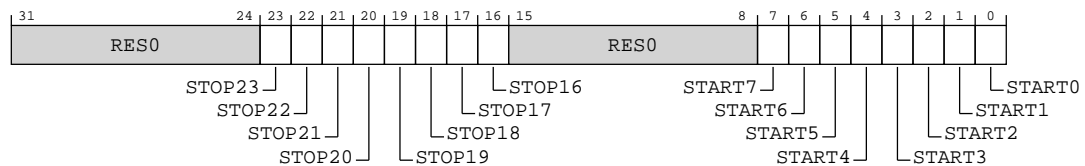
W

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-392: EXT_TRCVISSCTLR bit assignments****Table B-762: TRCVISSCTLR bit descriptions**

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:16]	STOP<m>, bit[m], where m = 23 to 16	<p>Selects which Single Address Comparators are in use with ViewInst start/stop function, for the purpose of stopping trace.</p> <p>0b0 The Single Address Comparator m, is not selected as a stop resource.</p> <p>0b1 The Single Address Comparator m, is selected as a stop resource.</p>	8 {x}
[15:8]	RES0	Reserved	RES0
[7:0]	START<m>, bit[m], where m = 7 to 0	<p>Selects which Single Address Comparators are in use with ViewInst start/stop function, for the purpose of starting trace.</p> <p>0b0 The Single Address Comparator m, is not selected as a start resource.</p> <p>0b1 The Single Address Comparator m, is selected as a start resource.</p>	8 {x}

Access

Must be programmed if ext-TRCIDR4.NUMACPAIRS > 0b0000.

For any 2 comparators selected for the ViewInst start/stop function, the comparator containing the lower address must be a lower numbered comparator.

Accessibility

Must be programmed if ext-TRCIDR4.NUMACPAIRS > 0b0000.

For any 2 comparators selected for the ViewInst start/stop function, the comparator containing the lower address must be a lower numbered comparator.

Component	Offset	Range
ETM	0x088	None

This interface is accessible as follows:

RW

B.2.2.7.16 TRCVDCTLR, ViewData Main Control Register

Controls data trace filtering.

Configurations

External register TRCVDCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.23 TRCVDCTLR, ViewData Main Control Register](#) on page 1265 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x0A0

Access type

Read

R

Write

W

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-393: EXT_TRCVDCTLR bit assignments

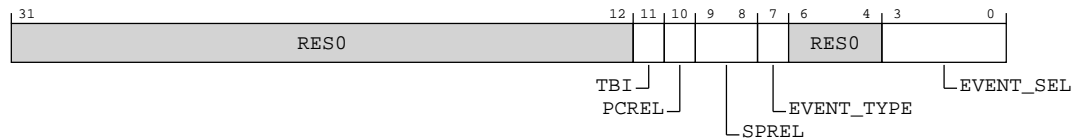


Table B-764: TRCVDCTLR bit descriptions

Bits	Name	Description	Reset
[31:12]	RES0	Reserved	RES0
[11]	TBI	Controls which information a trace unit populates in bits[63:56] of the data address. 0b0 The trace unit assigns bits[63:56] to have the same value as bit[55] of the data address, that is, it sign-extends the value. 0b1 The trace unit assigns bits[63:56] to have the same value as bits[63:56] of the data address.	x
[10]	PCREL	Controls whether a trace unit traces data for transfers that are relative to the Program Counter (PC). 0b0 The trace unit does not affect the tracing of PC-relative transfers. 0b1 The trace unit does not trace the address or value portions of PC-relative transfers. This bit only affects PC-relative transfers that use the PC as a base register plus an immediate offset.	x
[9:8]	SPREL	Controls whether a trace unit traces data for transfers that are relative to the Stack Pointer (SP). 0b00 The trace unit does not affect the tracing of SP-relative transfers. 0b01 Reserved. 0b10 The trace unit does not trace the address portion of SP-relative transfers. If data value tracing is enabled then the trace unit generates a P1 data address element. 0b11 The trace unit does not trace the address or value portions of SP-relative transfers.	xx

Bits	Name	Description	Reset
[7]	EVENT_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. EVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. EVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. EVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0
[3:0]	EVENT_SEL	Defines the selected Resource Selector or pair of Resource Selectors. EVENT_TYPE controls whether EVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire.	xxxx

Access

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when data tracing is enabled, that is, when either ext-TRCCONFIGR.DA==1 or ext-TRCCONFIGR.DV==1.

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when data tracing is enabled, that is, when either ext-TRCCONFIGR.DA==1 or ext-TRCCONFIGR.DV==1.

Component	Offset	Range
ETM	0x0A0	None

This interface is accessible as follows:

RW

B.2.2.7.17 TRCVDSACCTLR, ViewData Include/Exclude Single Address Comparator Control Register

Use this to select, or read, the Single Address Comparators for the ViewData include/exclude control.

Configurations

External register TRCVDSACCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.27 TRCVDSACCTLR, ViewData Include/Exclude Single Address Comparator Control Register](#) on page 1276 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x0A4

Access type

Read

R

Write

W

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-394: EXT_TRCVDSACCTLR bit assignments

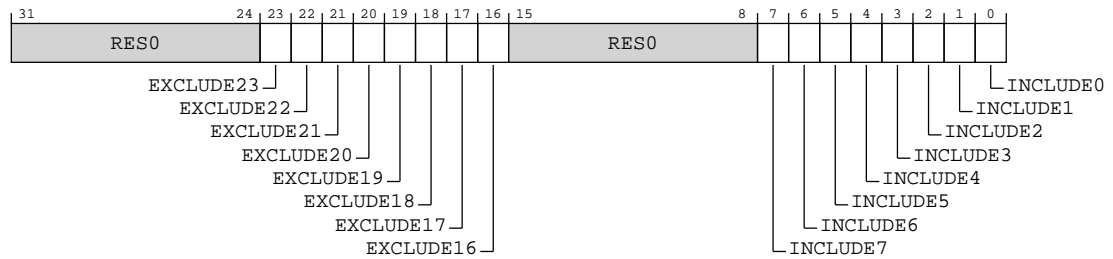


Table B-766: TRCVDSACCTLR bit descriptions

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	RES0
[23:16]	EXCLUDE<m>, bit[m], where m = 23 to 16	<p>Selects which single address comparators are in use with ViewData exclude control..</p> <p>0b0</p> <p>The single address comparator m, is not selected for exclude control.</p> <p>0b1</p> <p>The single address comparator m, is selected for exclude control.</p>	8{x}
[15:8]	RES0	Reserved	RES0
[7:0]	INCLUDE<m>, bit[m], where m = 7 to 0	<p>Selects which single address comparators are in use with ViewData include control.</p> <p>0b0</p> <p>The single address comparator m, is not selected for include control.</p> <p>0b1</p> <p>The single address comparator m, is selected for include control.</p> <p>If no single address comparators and no address range comparators are selected to be included, then all data transfers are included by default. The exclude control then indicates which data transfers are excluded.</p>	8{x}

Access

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, ext-TRCCONFIGR.DA==1 or ext-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, ext-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects a single address comparator that is either:

- Programmed to be sensitive to a data value comparator.
- Programmed to be an instruction address comparator.

In these situations, data transfers might be traced unexpectedly or might not be traced.

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, ext-TRCCONFIGR.DA==1 or ext-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, ext-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects a single address comparator that is either:

- Programmed to be sensitive to a data value comparator.
- Programmed to be an instruction address comparator.

In these situations, data transfers might be traced unexpectedly or might not be traced.

Component	Offset	Range
ETM	0x0A4	None

This interface is accessible as follows:

RW

B.2.2.7.18 TRCVDARCCTLR, ViewData Include/Exclude Address Range Comparator Control Register

Use this to select, or read, the Address Range Comparator for the ViewData include/exclude control.

Configurations

External register TRCVDARCCTLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.29 TRCVDARCCTLR, ViewData Include/Exclude Address Range Comparator Control Register](#) on page 1280 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x0A8

Access type

Read

R

Write

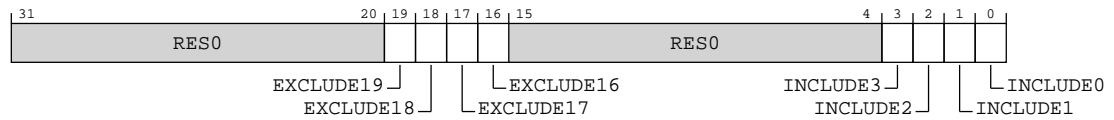
W

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-395: EXT_TRCVDARCCTLR bit assignments****Table B-768: TRCVDARCCTLR bit descriptions**

Bits	Name	Description	Reset
[31:20]	RES0	Reserved	RES0
[19:16]	EXCLUDE<m>, bit[m], where m = 19 to 16	<p>Exclude range field. Selects which address range comparator pairs are in use with ViewData exclude control.</p> <p>0b0</p> <p>The address range that address range comparator pair m defines, is not selected for ViewData exclude control.</p> <p>0b1</p> <p>The address range that address range comparator pair m defines, is selected for ViewData exclude control.</p>	xxxx
[15:4]	RES0	Reserved	RES0
[3:0]	INCLUDE<m>, bit[m], where m = 3 to 0	<p>Include range field. Selects which address range comparator pairs are in use with ViewData include control.</p> <p>0b0</p> <p>The address range that address range comparator pair m defines, is not selected for ViewData include control.</p> <p>0b1</p> <p>The address range that address range comparator pair m defines, is selected for ViewData include control.</p> <p>If no single address comparators and no address range comparators are selected to be included, then all data transfers are included by default. The exclude control then indicates which data transfers are excluded.</p>	xxxx

Access

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, ext-TRCCONFIGR.DA==1 or ext-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, ext-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects an address range comparator that is programmed to be sensitive to a data value comparator, data transfers might be traced unexpectedly or might not be traced.

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

This register must be programmed when both of the following apply:

- Data tracing is enabled, that is, ext-TRCCONFIGR.DA==1 or ext-TRCCONFIGR.DV==1.
- One or more address comparators are implemented, that is, ext-TRCIDR4.NUMACPAIRS > 0b0000.

If software writes to this register and selects an address range comparator that is programmed to be sensitive to a data value comparator, data transfers might be traced unexpectedly or might not be traced.

Component	Offset	Range
ETM	0x0A8	None

This interface is accessible as follows:

RW

B.2.2.7.19 TRCSEQEVR<n>, Sequencer State Transition Control Register <n> , n = 0 - 2

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

External register TRCSEQEVR<n> bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.1 TRCSEQEVR<n>, Sequencer State Transition Control Register <n> , n = 0 - 2](#) on page 1216 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x100 + (4 * n)

Access type

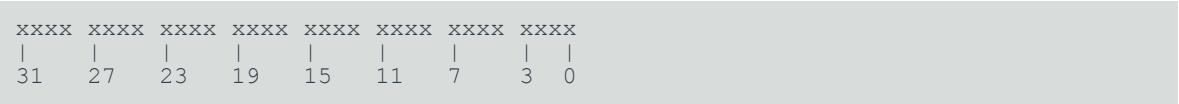
Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-396: EXT_TRCSEQEVR<n> bit assignments

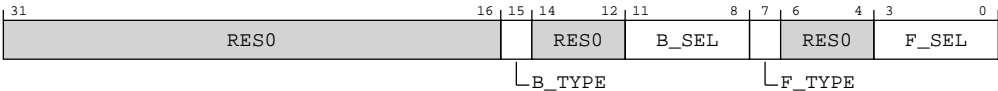


Table B-770: TRCSEQEVR<n> bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[15]	B_TYPE	<p>Chooses the type of Resource Selector.</p> <p>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.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>B_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>B_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. B_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[14:12]	RES0	Reserved	RES0
[11:8]	B_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. B_TYPE controls whether B_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>Backward field. Selects the single Resource Selector or Resource Selector pair.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx
[7]	F_TYPE	<p>Chooses the type of Resource Selector.</p> <p>Forward 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.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>F_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>F_SEL[2:0] 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. F_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[6:4]	RES0	Reserved	RES0
[3:0]	F_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. F_TYPE controls whether F_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>Forward field. Selects the single Resource Selector or Resource Selector pair.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Component	Offset	Range
ETM	0x100 + (4 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.20 TRCSEQRSTEV, Sequencer Reset Control Register

Moves the sequencer to state 0 when a programmed resource event occurs.

Configurations

External register TRCSEQRSTEV bits [31:0] are architecturally mapped to AArch64 System register A.2.8.20 TRCSEQRSTEV, Sequencer Reset Control Register on page 1257 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x118

Access type

Read

R

Write

W

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-397: EXT_TRCSEQRSTEVr bit assignments

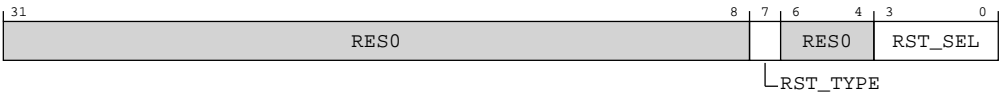


Table B-772: TRCSEQRSTEVr bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7]	RST_TYPE	Chooses the type of Resource Selector. 0b0 A single Resource Selector. RST_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event. 0b1 A Boolean-combined pair of Resource Selectors. RST_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. RST_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.	x
[6:4]	RES0	Reserved	RES0
[3:0]	RST_SEL	Defines the selected Resource Selector or pair of Resource Selectors. RST_TYPE controls whether RST_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors. If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE , and the resource event might fire or might not fire.	xxxx

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Component	Offset	Range
ETM	0x118	None

This interface is accessible as follows:

RW

B.2.2.7.21 TRCSEQSTR, Sequencer State Register

Use this to set, or read, the sequencer state.

Configurations

External register TRCSEQSTR bits [31:0] are architecturally mapped to AArch64 System register A.2.8.21 TRCSEQSTR, Sequencer State Register on page 1260 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x11C

Access type

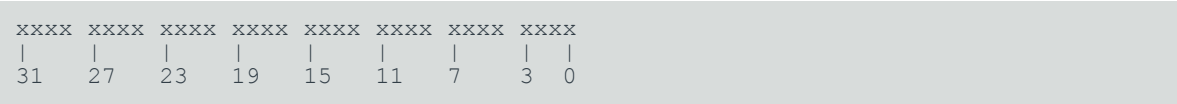
Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-398: EXT_TRCSEQSTR bit assignments

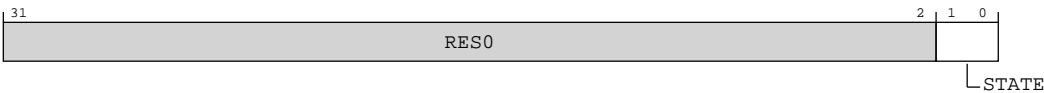


Table B-774: TRCSEQSTR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[1:0]	STATE	Set or returns the state of the sequencer. 0b00 State 0. 0b01 State 1. 0b10 State 2. 0b11 State 3.	xx

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.SEQUENCER != 0b0000.

Component	Offset	Range
ETM	0x11C	None

This interface is accessible as follows:

RW

B.2.2.7.22 TRCEXTINSEL, External Input Select Register

Use this to set, or read, which external inputs are resources to the trace unit.

Configurations

External register TRCEXTINSEL bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.24 TRCEXTINSEL, External Input Select Register](#) on page 1268 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x120

Access type

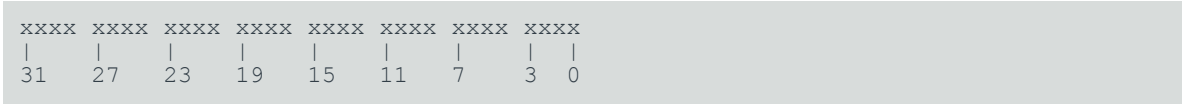
Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-399: EXT_TRCEXTINSEL bit assignments

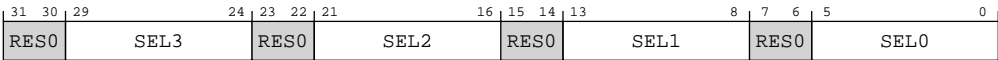


Table B-776: TRCEXTINSEL bit descriptions

Bits	Name	Description	Reset
[31:30]	RES0	Reserved	RES0
[29:24]	SEL3	Selects external input resource #3.	6 { x }
[23:22]	RES0	Reserved	RES0
[21:16]	SEL2	Selects external input resource #2.	6 { x }
[15:14]	RES0	Reserved	RES0
[13:8]	SEL1	Selects external input resource #1.	6 { x }
[7:6]	RES0	Reserved	RES0
[5:0]	SEL0	Selects external input resource #0.	6 { x }

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

Component	Offset	Range
ETM	0x120	None

This interface is accessible as follows:

RW

B.2.2.7.23 TRCCNTRLDVR<n>, Counter Reload Value Register <n> , n = 0 - 1

This sets or returns the reload count value for counter <n>.

Configurations

External register TRCCNTRLDVR<n> bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.2 TRCCNTRLDVR<n>, Counter Reload Value Register <n> , n = 0 - 1](#) on page 1219 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x140 + (4 * n)

Access type

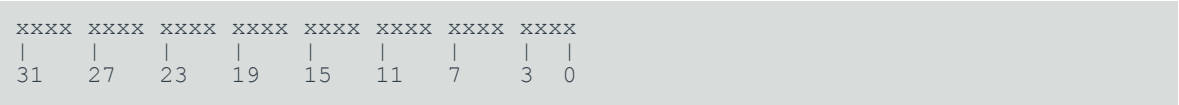
Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-400: EXT_TRCCNTRLDVR<n> bit assignments

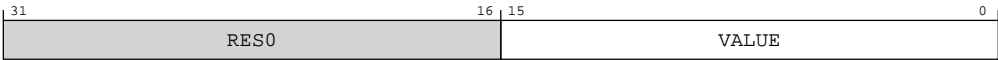


Table B-778: TRCCNTRLDVR<n> bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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}

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 0b1.

Component	Offset	Range
ETM	0x140 + (4 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.24 TRCCNTCTLR<n>, Counter Control Register <n> , n = 0 - 1

Controls the operation of counter <n>.

Configurations

External register TRCCNTCTLR<n> bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.3 TRCCNTCTLR<n>, Counter Control Register <n> , n = 0 - 1](#) on page 1221 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x150 + (4 * n)

Access type

Read

R

Write

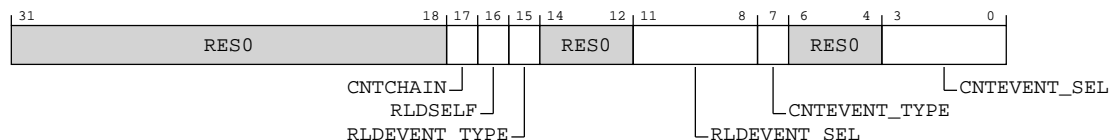
W

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-401: EXT_TRCCNTCTLR<n> bit assignments****Table B-780: TRCCNTCTLR<n> bit descriptions**

Bits	Name	Description	Reset
[31:18]	RES0	Reserved	RES0
[17]	CNTCHAIN	For TRCCNTCTLR1, this bit controls whether the counter decrements when a reload event occurs for counter <n-1>. 0b0 The counter does not decrement when a reload event for counter <n-1> occurs. 0b1 Counter <n> decrements when a reload event for counter <n-1> occurs. This concatenates counter <n> and counter <n-1>, to provide a larger count value. CNTCHAIN is not implemented for TRCCNTCTLR0.	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	<p>Chooses the type of Resource Selector.</p> <p>Selects an event, that when it occurs causes a reload event for counter <n>.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>RLDEVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>RLDEVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. RLDEVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[14:12]	RES0	Reserved	RES0
[11:8]	RLDEVENT_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. RLDEVENT_TYPE controls whether RLDEVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>Selects an event, that when it occurs causes a reload event for counter <n>.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx
[7]	CNTEVENT_TYPE	<p>Chooses the type of Resource Selector.</p> <p>Selects an event, that when it occurs causes counter <n> to decrement.</p> <p>0b0</p> <p>A single Resource Selector.</p> <p>CNTEVENT_SEL[3:0] selects the single Resource Selector, from 0-15, used to activate the resource event.</p> <p>0b1</p> <p>A Boolean-combined pair of Resource Selectors.</p> <p>CNTEVENT_SEL[2:0] selects the Resource Selector pair, from 0-7, that has a Boolean function that is applied to it whose output is used to activate the resource event. CNTEVENT_SEL[3] is ignored by the Trace unit and behaves as if it was programmed to 0.</p>	x
[6:4]	RES0	Reserved	RES0
[3:0]	CNTEVENT_SEL	<p>Defines the selected Resource Selector or pair of Resource Selectors. CNTEVENT_TYPE controls whether CNTEVENT_SEL is the index of a single Resource Selector, or the index of a pair of Resource Selectors.</p> <p>Selects an event, that when it occurs causes counter <n> to decrement.</p> <p>If an unimplemented Resource Selector is selected using this field, the behavior of the resource event is UNPREDICTABLE, and the resource event might fire or might not fire.</p>	xxxx

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 0b1.

Component	Offset	Range
ETM	0x150 + (4 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.25 TRCCNTVR<n>, Counter Value Register <n> , n = 0 - 1

This sets or returns the value of counter <n>.

Configurations

External register TRCCNTVR<n> bits [31:0] are architecturally mapped to AArch64 System register A.2.8.4 TRCCNTVR<n>, Counter Value Register <n> , n = 0 - 1 on page 1224 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x160 + (4 * n)

Access type

Read

R

Write

W

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-402: EXT_TRCCNTVR<n> bit assignments

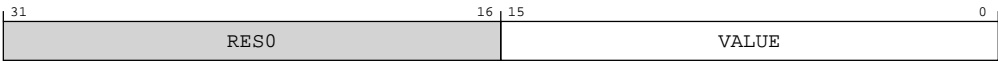


Table B-782: TRCCNTVR<n> bit descriptions

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	RES0
[15:0]	VALUE	Contains the count value of counter.	1 6 { x }

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0010 and TRCRSCTLR<a>.COUNTERS[n] == 0b1.

Component	Offset	Range
ETM	0x160 + (4 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.26 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

Configurations

External register TRCIDR8 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.7 TRCIDR8, ID Register 8](#) on page 1232 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x180

Access type

RO

Reset value

0000 0000 0000 0000 0000 0000 0000 0001

Bit descriptions

Figure B-403: EXT_TRCIDR8 bit assignments

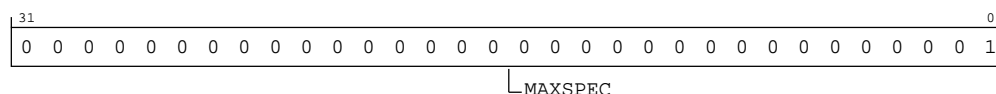


Table B-784: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[31:0]	MAXSPEC	<p>Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of PO elements in the trace element stream that can be speculative at any time.</p> <p>0b00000000000000000000000000000001</p> <p>1 speculative PO element.</p>	0x00000001

Accessibility

Component	Offset	Range
ETM	0x180	None

This interface is accessible as follows:

RO

B.2.2.7.27 TRCIDR9, ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR9 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.11 TRCIDR9, ID Register 9](#) on page 1240 bits [31:0].

Attributes

Width

32

Component

FTM

Register offset

0x184

Access type

RO

Reset value

0000 0000 0000 0000 0000 0000 0010 0000

Bit descriptions

Figure B-404: EXT_TRCIDR9 bit assignments

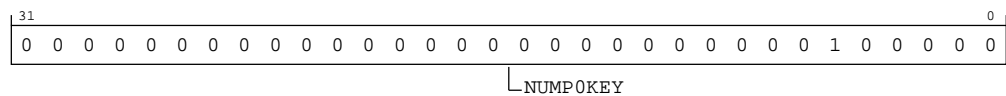


Table B-786: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMPOKEY	Indicates the number of PO right-hand keys. 0b0000000000000000000000000000100000 32 PO right-hand keys.	0x00000020

Accessibility

Component	Offset	Range
ETM	0x184	None

This interface is accessible as follows:

RO

B.2.2.7.28 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR10 bits [31:0] are architecturally mapped to AArch64 System register A.2.8.13 TRCIDR10, ID Register 10 on page 1244 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x188

Access type

RO

Reset value

0000 0000 0000 0000 0000 0000 0000 1000

Bit descriptions

Figure B-405: EXT_TRCIDR10 bit assignments

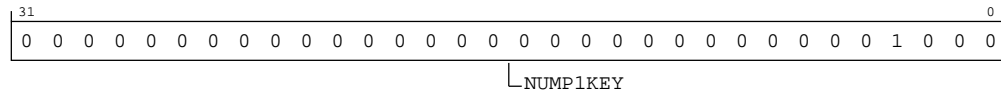


Table B-788: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMP1KEY	Indicates the number of P1 right-hand keys. 0b000000000000000000000000000000001000 8 P1 right-hand keys.	0x00000008

Accessibility

Component	Offset	Range
ETM	0x188	None

This interface is accessible as follows:

RO

B.2.2.7.29 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR11 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.15 TRCIDR11, ID Register 11](#) on page 1247 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x18C

Access type

RO

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-406: EXT_TRCIDR11 bit assignments

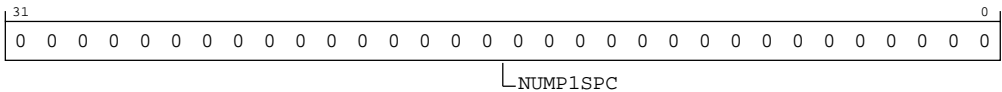


Table B-790: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMP1SPC	Indicates the number of special P1 right-hand keys. 0b00000000000000000000000000000000 No special P1 right-hand keys.	0x00000000

Accessibility

Component	Offset	Range
ETM	0x18C	None

This interface is accessible as follows:

RO

B.2.2.7.30 TRCIDR12, ID Register 12

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR12 bits [31:0] are architecturally mapped to AArch64 System register A.2.8.17 TRCIDR12, ID Register 12 on page 1252 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x190

Access type

RO

Reset value

0000 0000 0000 0000 0000 0000 0000 0001

Bit descriptions

Figure B-407: EXT_TRCIDR12 bit assignments

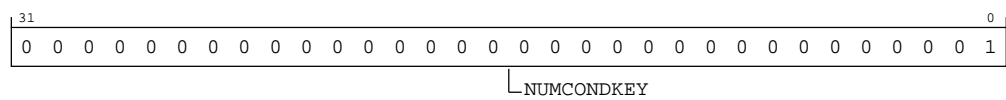


Table B-792: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMCONDKEY	Indicates the number of conditional instruction right-hand keys. 0b00000000000000000000000000000001 1 conditional instruction right-hand key.	0x00000001

Accessibility

Component	Offset	Range
ETM	0x190	None

This interface is accessible as follows:

RO

B.2.2.7.31 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR13 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.18 TRCIDR13, ID Register 13](#) on page 1254 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x194

Access type

RO

Reset value

0000 0000 0000 0000 0000 0000 0000 0000

Bit descriptions

Figure B-408: EXT_TRCIDR13 bit assignments

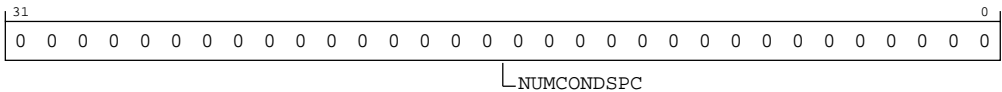


Table B-794: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMCONDSPC	Indicates the number of special conditional instruction right-hand keys. 0b00000000000000000000000000000000 No special conditional instruction right-hand keys.	0x00000000

Accessibility

Component	Offset	Range
ETM	0x194	None

This interface is accessible as follows:

RO

B.2.2.7.32 TRCIMSPECO, IMP DEF Register 0

TRCIMSPECO shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

Configurations

External register TRCIMSPECO bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.8 TRCIMSPECO, IMP DEF Register 0](#) on page 1234 bits [31:0].

Attributes

Width

32

Component

ETM

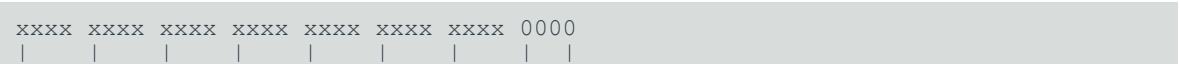
Register offset

0x1C0


Access type

RO

Reset value



312723191511730



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-409: EXT_TRCIMSPEC0 bit assignments

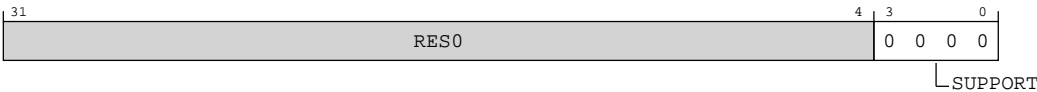


Table B-796: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	SUPPORT	Indicates whether the implementation supports IMPLEMENTATION DEFINED features. 0b0000 No IMPLEMENTATION DEFINED features are supported.	0b0000

Accessibility

Component	Offset	Range
ETM	0x1C0	None

This interface is accessible as follows:

RW

B.2.2.7.33 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR0 bits [31:0] are architecturally mapped to AArch64 System register A.2.8.25 TRCIDR0, ID Register 0 on page 1270 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x1E0

Access type

RO

Reset value

xx00	1000	xxxx	xx00	0x00	111x	1111	111x
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-410: EXT_TRCIDR0 bit assignments

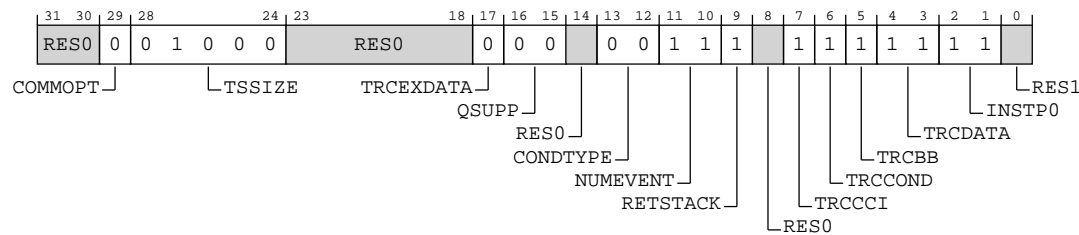


Table B-798: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:30]	RES0	Reserved	RES0
[29]	COMMOPT	Indicates how the commit field in Cycle count packets is interpreted. 0b0 Commit mode 0. Cycle count packets includes the Commit element to which the Cycle Count element is attached.	0b0
[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
[23:18]	RES0	Reserved	RES0
[17]	TRCEXDATA	Indicates if the trace unit implements tracing of data transfers for exceptions and exception returns. 0b0 Tracing of data transfers for exceptions and exception returns not implemented.	0b0

Bits	Name	Description	Reset
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. 0b00 Q element support is not implemented.	0b00
[14]	RES0	Reserved	RES0
[13:12]	CONDTYPE	Indicates how conditional instructions are traced. 0b00 Conditional instructions are traced with an indication of whether they pass or fail their condition code check.	0b00
[11:10]	NUMEVENT	Indicates the number of trace events implemented. 0b11 The trace unit supports 4 trace events.	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. 0b1 Conditional instruction tracing implemented.	0b1
[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. 0b11 Data tracing implemented.	0b11
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. 0b11 Load and store instructions are P0 instructions.	0b11
[0]	RES1	Reserved	RES1

Accessibility

Component	Offset	Range
ETM	0x1E0	None

This interface is accessible as follows:

RO

B.2.2.7.34 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.28 TRCIDR1, ID Register 1](#) on page 1279 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x1E4

Access type

RO

Reset value

0100	0001	xxxx	xxxx	xxxx	0100	0101	0111
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-411: EXT_TRCIDR1 bit assignments

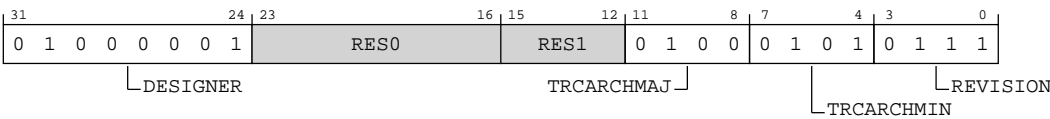


Table B-800: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. 0b01000001 Arm Limited.	0x41
[23:16]	RES0	Reserved	RES0
[15:12]	RES1	Reserved	RES1

Bits	Name	Description	Reset
[11:8]	TRCARCHMAJ	Major architecture version. 0b0100 ETMv4.	0b0100
[7:4]	TRCARCHMIN	Minor architecture version. 0b0101 ETMv4.5. All other values are reserved.	0b0101
[3:0]	REVISION	Implementation revision. 0b0111 Revision 7.	0b0111

Accessibility

Component	Offset	Range
ETM	0x1E4	None

This interface is accessible as follows:

RO

B.2.2.7.35 TRCIDR2, ID Register 2

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR2 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.30 TRCIDR2, ID Register 2](#) on page 1283 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x1E8

Access type

RO

Reset value

1100 0000 1000 0100 0001 0000 1000 1000

Bit descriptions

Figure B-412: EXT_TRCIDR2 bit assignments

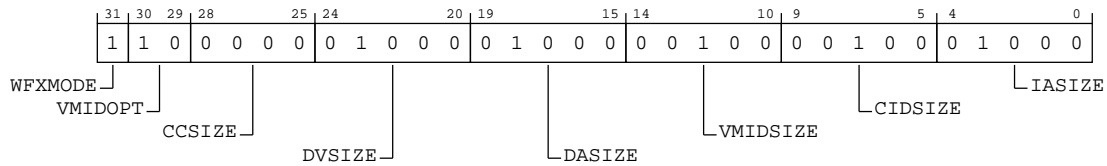


Table B-802: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as PO instructions: 0b1 WFI and WFE instructions are classified as PO instructions.	0b1
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. 0b10 Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is RES1 .	0b10
[28:25]	CCSIZE	Indicates the size of the cycle counter. 0b0000 The cycle counter is 12 bits in length.	0b0000
[24:20]	DVSIZE	Indicates the data value size in bytes. 0b01000 Data value tracing has a maximum of 64-bit data values.	0b01000
[19:15]	DASIZE	Indicates the data address size in bytes. 0b01000 Data address tracing has a maximum of 64-bit data addresses.	0b01000
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. 0b00100 32-bit Virtual context identifier size.	0b00100
[9:5]	CIDSIZE	Indicates the Context identifier size. 0b00100 32-bit Context identifier size.	0b00100
[4:0]	IASIZE	Virtual instruction address size. 0b01000 Maximum of 64-bit instruction address size.	0b01000

Accessibility

Component	Offset	Range
ETM	0x1E8	None

This interface is accessible as follows:

RO

B.2.2.7.36 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

Configurations

External register TRCIDR3 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.32 TRCIDR3, ID Register 3](#) on page 1288 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x1EC

Access type

RO

Reset value

x000	1101	x000	0111	xx00	0000	0000	0100
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-413: EXT_TRCIDR3 bit assignments

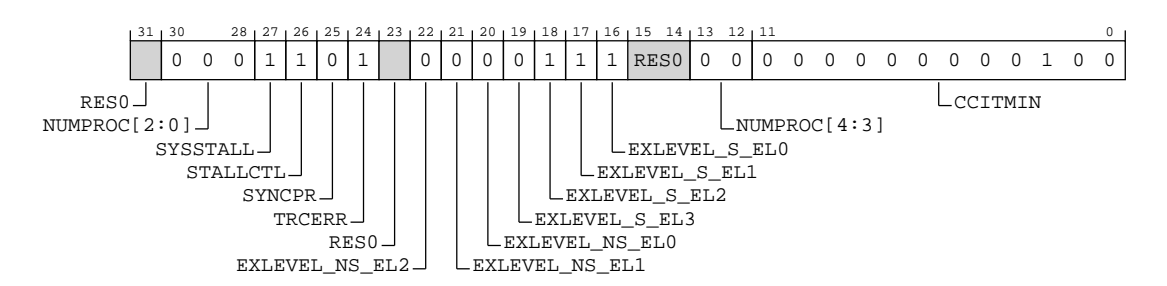


Table B-804: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. 0b1 Stalling of the PE is permitted.	0b1
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. 0b1 Stalling of the PE is implemented.	0b1
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. 0b0 ext-TRCSYNCPR is read-write so software can change the synchronization period.	0b0
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. 0b1 Forced tracing of System Error exceptions is implemented.	0b1
[23]	RES0	Reserved	RES0
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. 0b0 Non-secure EL2 is not implemented.	0b0
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. 0b0 Non-secure EL1 is not implemented.	0b0
[20]	EXLEVEL_NS_ELO	Indicates if Non-secure ELO implemented. 0b0 Non-secure ELO is not implemented.	0b0
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. 0b0 Secure EL3 is not implemented.	0b0
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. 0b1 Secure EL2 is implemented.	0b1
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. 0b1 Secure EL1 is implemented.	0b1
[16]	EXLEVEL_S_ELO	Indicates if Secure ELO implemented. 0b1 Secure ELO 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. This field reads as 0b00000.	0b00000

Bits	Name	Description	Reset
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in ext-TRCCCCTLR.THRESHOLD. 0b000000000100 Minimum allowed threshold value is 4.	0x004

Accessibility

Component	Offset	Range
ETM	0x1EC	None

This interface is accessible as follows:

RO

B.2.2.7.37 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR4 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.34 TRCIDR4, ID Register 4](#) on page 1293 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x1F0

Access type

RO

Reset value

0001	0001	0010	0111	0000	xxx1	0010	0100
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-414: EXT_TRCIDR4 bit assignments

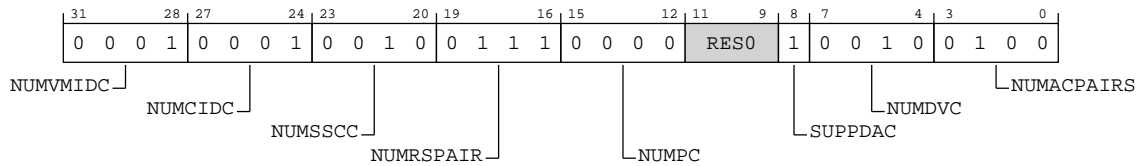


Table B-806: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. 0b0001 The implementation has one Virtual Context Identifier Comparator.	0b0001
[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. 0b0010 The implementation has two Single-shot Comparator Controls.	0b0010
[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. 0b1 Data address comparisons implemented.	0b1
[7:4]	NUMDVC	Indicates the number of data value comparators. 0b0010 Two data value comparators implemented.	0b0010
[3:0]	NUMACPAIRS	Indicates the number of address comparator pairs that are available for tracing. 0b0100 The implementation has four address comparator pairs.	0b0100

Accessibility

Component	Offset	Range
ETM	0x1F0	None

This interface is accessible as follows:

RO

B.2.2.7.38 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR5 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.36 TRCIDR5, ID Register 5](#) on page 1298 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x1F4

Access type

RO

Reset value

0010	100x	1100	0111	xxxx	1000	0011	1010
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-415: EXT_TRCIDR5 bit assignments

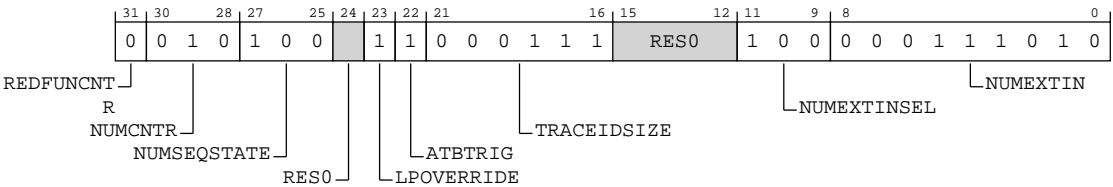


Table B-808: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[31]	REDFUNCNTR	Indicates if the reduced function counter is implemented. 0b0 The reduced function counter is not supported.	0b0
[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. 0b1 The trace unit support Low-power Override Mode.	0b1
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. 0b1 The implementation supports ATB triggers.	0b1
[21:16]	TRACEIDSIZE	Indicates the trace ID width. 0b000111 The implementation supports a 7-bit trace ID.	0b000111
[15:12]	RES0	Reserved	RES0
[11:9]	NUMEXTINSEL	Indicates how many external input selector resources are implemented. 0b100 4 external input selector resources are available.	0b100
[8:0]	NUMEXTIN	Indicates how many external inputs are implemented. 0b000111010 The implementation has 58 external inputs.	0b000111010

Accessibility

Component	Offset	Range
ETM	0x1F4	None

This interface is accessible as follows:

RO

B.2.2.7.39 TRCIDR6, ID Register 6

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR6 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.38 TRCIDR6, ID Register 6](#) on page 1302 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x1F8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-416: EXT_TRCIDR6 bit assignments

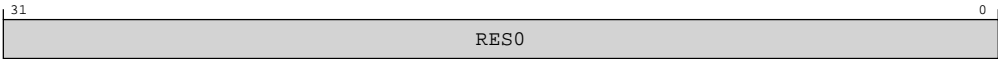


Table B-810: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0x1F8	None

This interface is accessible as follows:

RO

B.2.2.7.40 TRCIDR7, ID Register 7

Returns the tracing capabilities of the trace unit.

Configurations

External register TRCIDR7 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.40 TRCIDR7, ID Register 7](#) on page 1306 bits [31:0].

Attributes

Width

32

Component

ETM

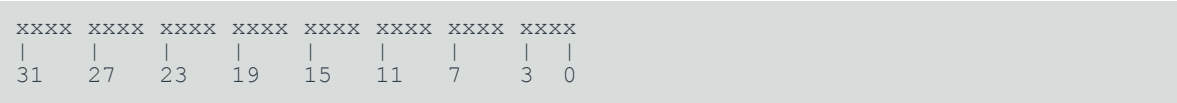
Register offset

0x1FC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-417: EXT_TRCIDR7 bit assignments

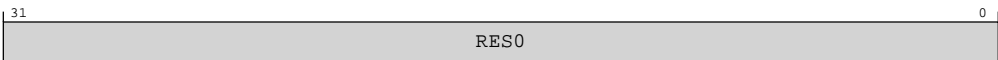


Table B-812: TRCIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0x1FC	None

This interface is accessible as follows:

RO

B.2.2.7.41 TRCRSCTLR<n>, Resource Selection Control Register <n> , n = 2 - 15

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.

External register TRCRSCTLR<n> bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.45 TRCRSCTLR<n>, Resource Selection Control Register <n> , n = 2 - 15](#) on page 1316 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x200 + (4 * n)

Access type

Read

R

Write

W

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-418: EXT_TRCRSCTLR<n> bit assignments

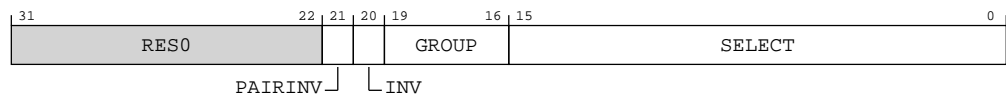


Table B-814: TRCRSCTLR<n> bit descriptions

Bits	Name	Description	Reset
[31:22]	RES0	Reserved	RES0
[21]	PAIRINV	<div>When n MOD 2 == 0 For TRCRSCTLR<n>, where n is even, 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. Otherwise RES0</div>	x

Bits	Name	Description	Reset
[20]	INV	<p>Controls whether the resource, that GROUP and SELECT selects, is inverted.</p> <p>0b0 Do not invert the output of this selector.</p> <p>0b1 Invert the output of this selector.</p> <p>If:</p> <ul style="list-style-type: none"> A is the register TRCRSCTLR<m> where m is even. B is the register TRCRSCTLR<m+1>. <p>Then the combined output of the 2 resource selectors A and B depends on the value of (A.PAIRINV, A.INV, B.INV) as follows:</p> <ul style="list-style-type: none"> 0b000 -> A and B. 0b001 -> RESERVED. 0b010 -> not(A) and B. 0b011 -> not(A) and not(B). 0b100 -> not(A) or not(B). 0b101 -> not(A) or B. 0b110 -> RESERVED. 0b111 -> A or B. 	x
[19:16]	GROUP	<p>Selects a group of resources.</p> <p>0b0000 External input selectors.</p> <p>0b0001 PE Comparator Inputs.</p> <p>0b0010 Counters and Sequencer.</p> <p>0b0011 Single-shot Comparator Controls.</p> <p>0b0100 Single Address Comparators.</p> <p>0b0101 Address Range Comparators.</p> <p>0b0110 Context Identifier Comparators.</p> <p>0b0111 Virtual Context Identifier Comparators.</p> <p>All other values are reserved.</p>	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 15:0 Reserved, RES0.	16 {x}
[15] continued	SELECT	SELECT encoding for Counters and Sequencer 15:8 Reserved, RES0. SEQUENCER<m>, bit[m], for m = 7 to 4 Sequencer states. 0b0 Ignore Sequencer state m. 0b1 Select Sequencer state m. 3:2 Reserved, RES0. COUNTERS<m>, bit[m], for m = 1 to 0 Counters resources at zero. 0b0 Ignore Counter m. 0b1 Select Counter m is zero.	16 {x}

Bits	Name	Description	Reset
[15] continued	SELECT	SELECT encoding for Single-shot Comparator Controls 15:2 Reserved, RES0. SINGLE_SHOT<m>, bit[m], for m = 1 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 15:8 Reserved, RES0. SAC<m>, bit[m], for m = 7 to 0 Selects one or more Single Address Comparators. 0b0 Ignore Single Address Comparator m. 0b1 Select Single Address Comparator m.	16 {x}
[15] continued	SELECT	SELECT encoding for Address Range Comparators 15:4 Reserved, RES0. ARC<m>, bit[m], for m = 3 to 0 Selects one or more Address Range Comparators. 0b0 Ignore Address Range Comparator m. 0b1 Select Address Range Comparator m. SELECT encoding for Context Identifier Comparators 15:1 Reserved, RES0. CID<m>, bit[m], for m = 0 Selects one or more Context Identifier Comparators. 0b0 Ignore Context Identifier Comparator m. 0b1 Select Context Identifier Comparator m.	16 {x}

Bits	Name	Description	Reset
[15] continued	SELECT	SELECT encoding for Virtual Context Identifier Comparators 15:1 Reserved, RES0 . VMID<m>, bit[m], for m = 0 Selects one or more Virtual Context Identifier Comparators. 0b0 Ignore Virtual Context Identifier Comparator m. 0b1 Select Virtual Context Identifier Comparator m.	16 {x}

Access

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT0.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT0.SEL == n.
- ext-TRCEVENTCTLOR.EVENT0.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT0.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT1.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT1.SEL == n.
- ext-TRCEVENTCTLOR.EVENT1.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT1.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT2.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT2.SEL == n.
- ext-TRCEVENTCTLOR.EVENT2.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT2.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT3.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT3.SEL == n.
- ext-TRCEVENTCTLOR.EVENT3.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0b0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 0b1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0b0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 0b1 and TRCSEQEVR<a>.F.SEL = n/2.
- ext-TRCSEQRSTEV.RST.TYPE == 0b0 and ext-TRCSEQRSTEV.RST.SEL == n.

- ext-TRCSEQRSTEV.RST.TYPE == 0b1 and ext-TRCSEQRSTEV.RST.SEL == n/2.
- ext-TRCTSCTLR.EVENT.TYPE == 0b0 and ext-TRCTSCTLR.EVENT.SEL == n.
- ext-TRCTSCTLR.EVENT.TYPE == 0b1 and ext-TRCTSCTLR.EVENT.SEL == n/2.
- ext-TRCVICTLR.EVENT.TYPE == 0b0 and ext-TRCVICTLR.EVENT.SEL == n.
- ext-TRCVICTLR.EVENT.TYPE == 0b1 and ext-TRCVICTLR.EVENT.SEL == n/2.

Accessibility

Must be programmed if any of the following are true:

- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.RLDEVENT.SEL == n.
- TRCCNTCTLR<a>.RLDEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.RLDEVENT.SEL == n/2.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b0 and TRCCNTCTLR<a>.CNTEVENT.SEL == n.
- TRCCNTCTLR<a>.CNTEVENT.TYPE == 0b1 and TRCCNTCTLR<a>.CNTEVENT.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT0.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT0.SEL == n.
- ext-TRCEVENTCTLOR.EVENT0.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT0.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT1.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT1.SEL == n.
- ext-TRCEVENTCTLOR.EVENT1.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT1.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT2.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT2.SEL == n.
- ext-TRCEVENTCTLOR.EVENT2.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT2.SEL == n/2.
- ext-TRCEVENTCTLOR.EVENT3.TYPE == 0b0 and ext-TRCEVENTCTLOR.EVENT3.SEL == n.
- ext-TRCEVENTCTLOR.EVENT3.TYPE == 0b1 and ext-TRCEVENTCTLOR.EVENT3.SEL == n/2.
- TRCSEQEVR<a>.B.TYPE == 0b0 and TRCSEQEVR<a>.B.SEL = n.
- TRCSEQEVR<a>.B.TYPE == 0b1 and TRCSEQEVR<a>.B.SEL = n/2.
- TRCSEQEVR<a>.F.TYPE == 0b0 and TRCSEQEVR<a>.F.SEL = n.
- TRCSEQEVR<a>.F.TYPE == 0b1 and TRCSEQEVR<a>.F.SEL = n/2.
- ext-TRCSEQRSTEV.RST.TYPE == 0b0 and ext-TRCSEQRSTEV.RST.SEL == n.
- ext-TRCSEQRSTEV.RST.TYPE == 0b1 and ext-TRCSEQRSTEV.RST.SEL == n/2.
- ext-TRCTSCTLR.EVENT.TYPE == 0b0 and ext-TRCTSCTLR.EVENT.SEL == n.
- ext-TRCTSCTLR.EVENT.TYPE == 0b1 and ext-TRCTSCTLR.EVENT.SEL == n/2.
- ext-TRCVICTLR.EVENT.TYPE == 0b0 and ext-TRCVICTLR.EVENT.SEL == n.
- ext-TRCVICTLR.EVENT.TYPE == 0b1 and ext-TRCVICTLR.EVENT.SEL == n/2.

Component	Offset	Range
ETM	0x200 + (4 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.42 TRCSSCCR<n>, Single-shot Comparator Control Register <n> , n = 0 - 1

Controls the corresponding Single-shot Comparator Control resource.

Configurations

External register TRCSSCCR<n> bits [31:0] are architecturally mapped to AArch64 System register A.2.8.41 TRCSSCCR<n>, Single-shot Comparator Control Register <n> , n = 0 - 1 on page 1307 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x280 + (4 * n)

Access type

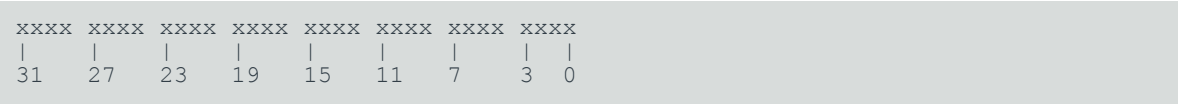
Read

R

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-419: EXT_TRCSSCCR<n> bit assignments

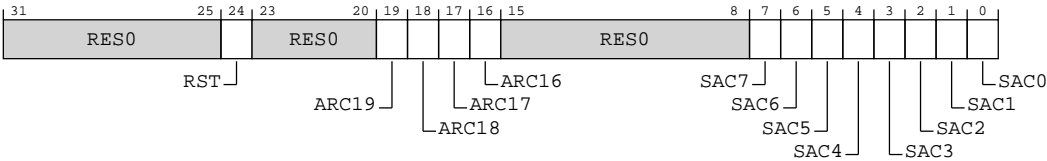


Table B-816: TRCSSCCR<n> bit descriptions

Bits	Name	Description	Reset
[31:25]	RES0	Reserved	RES0
[24]	RST	<p>Selects the Single-shot Comparator Control mode.</p> <p>0b0 The Single-shot Comparator Control is in single-shot mode.</p> <p>0b1 The Single-shot Comparator Control is in multi-shot mode.</p>	x
[23:20]	RES0	Reserved	RES0
[19:16]	ARC<m>, bit[m], where m = 19 to 16	<p>Selects one or more Address Range Comparators for Single-shot control.</p> <p>0b0 The Address Range Comparator m, is not selected for Single-shot control.</p> <p>0b1 The Address Range Comparator m, is selected for Single-shot control.</p>	xxxx
[15:8]	RES0	Reserved	RES0
[7:0]	SAC<m>, bit[m], where m = 7 to 0	<p>Selects one or more Single Address Comparators for Single-shot control.</p> <p>0b0 The Single Address Comparator m, is not selected for Single-shot control.</p> <p>0b1 The Single Address Comparator m, is selected for Single-shot control.</p>	8 {x}

Accessibility

Must be programmed if any TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 0b1.

Component	Offset	Range
ETM	0x280 + (4 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.43 TRCSSCSR<n>, Single-shot Comparator Control Status Register <n> , n = 0 - 1

Returns the status of the corresponding Single-shot Comparator Control.

Configurations

External register TRCSSCSR<n> bits [31:0] are architecturally mapped to AArch64 System register A.2.8.42 TRCSSCSR<n>, Single-shot Comparator Control Status Register <n> , n = 0 - 1 on page 1310 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x2A0 + (4 * n)

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-420: EXT_TRCSSCSR<n> bit assignments

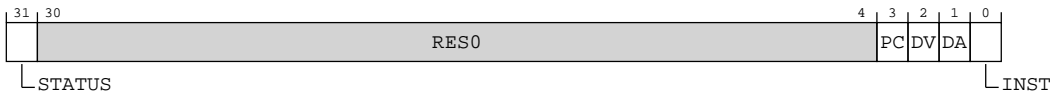


Table B-818: TRCSSCSR<n> bit descriptions

Bits	Name	Description	Reset
[31]	STATUS	<p>Single-shot Comparator Control status. Indicates if any of the comparators selected by this Single-shot Comparator control have matched. The selected comparators are defined by ext-TRCSSCCR<n>.ARC and ext-TRCSSCCR<n>.SAC.</p> <p>0b0</p> <p>No match has occurred. When the first match occurs, this field takes a value of 0b1. It remains at 0b1 until explicitly modified by a write to this register.</p> <p>0b1</p> <p>One or more matches has occurred. If ext-TRCSSCCR<n>.RST == 0b0 then:</p> <ul style="list-style-type: none">There is only one match and no more matches are possible.Software must reset this bit to 0b0 to re-enable the Single-shot Comparator Control. <p>The reset value is UNKNOWN. STATUS must be written to set an initial state when programming the trace unit, if the single-shot comparator is to be used.</p>	x
[30:4]	RES0	Reserved	RES0
[3]	PC	<p>PE Comparator Input support.</p> <p>0b0</p> <p>This Single-shot Comparator Control does not support PE Comparator Inputs.</p> <p>Access to this field is: RO</p>	x

Bits	Name	Description	Reset
[2]	DV	<p>When n == 0 Data value comparator support.</p> <p>0 This Single-shot Comparator Control does not support data value comparisons. TRCSSCSR0 has this value.</p> <p>When n == 1 Data value comparator support.</p> <p>1 This Single-shot Comparator Control supports data value comparisons. TRCSSCSR1 has this value.</p> <p>Otherwise RES0</p>	x
[1]	DA	<p>When n == 0 Data address comparator support.</p> <p>0 This Single-shot Comparator Control does not support data address comparisons. TRCSSCSR0 has this value.</p> <p>When n == 1 Data address comparator support.</p> <p>1 This Single-shot Comparator Control supports data address comparisons. TRCSSCSR1 has this value.</p> <p>Otherwise RES0</p>	x
[0]	INST	<p>When n == 0 Instruction address comparator support.</p> <p>1 This Single-shot Comparator Control supports instruction address comparisons. TRCSSCSR0 has this value.</p> <p>When n == 1 Instruction address comparator support.</p> <p>0 This Single-shot Comparator Control does not support instruction address comparisons. TRCSSCSR1 has this value.</p> <p>Otherwise RES0</p>	x

Accessibility

Must be programmed if TRCRSCTLR<a>.GROUP == 0b0011 and TRCRSCTLR<a>.SINGLE_SHOT[n] == 0b1.

Component	Offset	Range
ETM	0x2A0 + (4 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.44 TRCOSLAR, Trace OS Lock Access Register

Controls whether the Trace OS Lock is locked.

Configurations

External register TRCOSLAR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.43 TRCOSLAR, Trace OS Lock Access Register](#) on page 1313 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x300

Access type

RESERVEDW

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-421: EXT_TRCOSLAR bit assignments

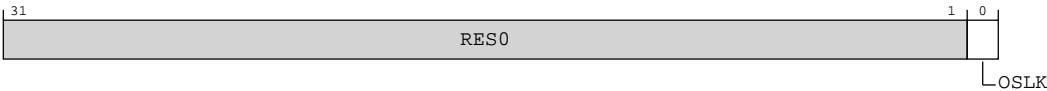


Table B-820: TRCOSLAR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	OSLK	OS Lock control bit. 0b0 Unlocks the OS Lock. 0b1 Locks the OS Lock. This setting disables the trace unit.	x

Accessibility

Component	Offset	Range
ETM	0x300	None

This interface is accessible as follows:

WO

B.2.2.7.45 TRCOSLSR, Trace OS Lock Status Register

Returns the status of the Trace OS Lock.

Configurations

External register TRCOSLSR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.44 TRCOSLSR, Trace OS Lock Status Register](#) on page 1314 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x304

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1x10
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-422: EXT_TRCOSLSR bit assignments

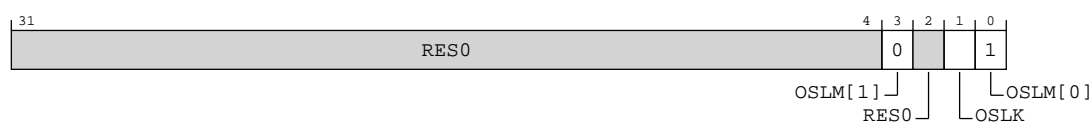


Table B-822: TRCOSLSR bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3, 0]	OSLM	OS Lock model. 0b10 Trace OS Lock is implemented. Access to this field is: RO	0b10
[2]	RES0	Reserved	RES0
[1]	OSLK	OS Lock status. 0b0 The OS Lock is unlocked. 0b1 The OS Lock is locked.	0b1

Accessibility

Component	Offset	Range
ETM	0x304	None

This interface is accessible as follows:

RO

B.2.2.7.46 TRCPDCR, PowerDown Control Register

Legacy power control register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0x310

Access type

RW

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-423: EXT_TRCPDCR bit assignments

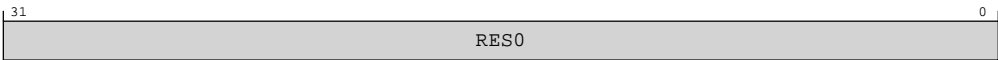


Table B-824: TRCPDCR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0x310	None

This interface is accessible as follows:

RW

B.2.2.7.47 TRCPDSR, PowerDown Status Register

Legacy power status register.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

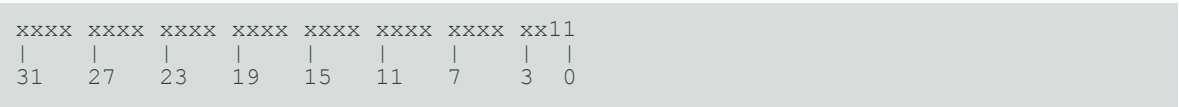
Register offset

0x314

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-424: EXT_TRCPDSR bit assignments

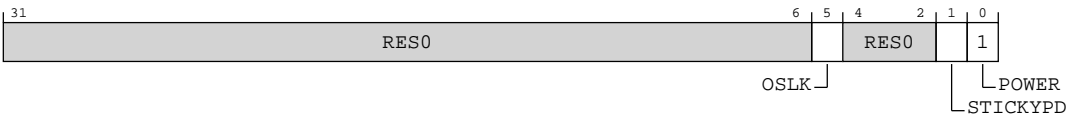


Table B-826: TRCPDSR bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	RES0
[5]	OSLK	OS Lock Status. 0b0 The OS Lock is unlocked. 0b1 The OS Lock is locked.	x
[4:2]	RES0	Reserved	RES0
[1]	STICKYPD	Sticky powerdown status. Indicates whether the trace register state is valid. 0b0 The state of ext-TRCOSLSR and the trace registers are valid. 0b1 The state of ext-TRCOSLSR and the trace registers might not be valid. This field is set to 1 if the power to the trace unit core power domain is removed and the trace unit register state is not valid. The STICKYPD field is read-sensitive. On a read of the TRCPDSR, this field is cleared to 0 after the register has been read.	0b1

Bits	Name	Description	Reset
[0]	POWER	Power Status. 0b1 The trace unit core power domain is powered. Trace unit registers are accessible.	0b1

Accessibility

Component	Offset	Range
ETM	0x314	None

This interface is accessible as follows:

RO

B.2.2.7.48 TRCACVR<n>, Address Comparator Value Register <n> , n = 0 - 7

Contains the address value.

Configurations

External register TRCACVR<n> bits [63:0] are architecturally mapped to AArch64 System register [A.2.8.48 TRCACVR<n>, Address Comparator Value Register <n> , n = 0 - 7](#) on page 1327 bits [63:0].

Attributes

Width

64

Component

ETM

Register offset

0x400 + (8 * n)

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	0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-425: EXT_TRCACVR<n> bit assignments

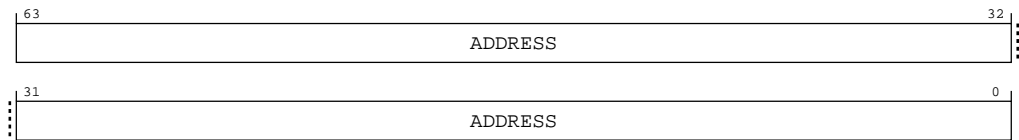


Table B-828: TRCACVR<n> bit descriptions

Bits	Name	Description	Reset
[63:0]	ADDRESS	<p>Address Value.</p> <p>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>.</p> <p>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 virtual address size supported by the PE.</p> <p>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.</p>	64 { x }

Access

Must be programmed if any of the following are true:

- ext-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- ext-TRCVIIECTLR.EXCLUDE[n/2] == 0b1.
- ext-TRCVIIECTLR.INCLUDE[n/2] == 0b1.
- ext-TRCVISSCTLR.START[n] == 0b1.
- ext-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.

- TRCSSCCR<>.SAC[n] == 0b1.
- ext-TRCQCTLR.RANGE[n/2] == 0b1.

Accessibility

Must be programmed if any of the following are true:

- ext-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- ext-TRCVIICTLR.EXCLUDE[n/2] == 0b1.
- ext-TRCVIICTLR.INCLUDE[n/2] == 0b1.
- ext-TRCVISSCTLR.START[n] == 0b1.
- ext-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.
- TRCSSCCR<>.SAC[n] == 0b1.
- ext-TRCQCTLR.RANGE[n/2] == 0b1.

Component	Offset	Range
ETM	0x400 + (8 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.49 TRCACATR<n>, Address Comparator Access Type Register <n> , n = 0 - 7

Defines the type of access for the corresponding ext-TRCACVTR<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

External register TRCACATR<n> bits [63:0] are architecturally mapped to AArch64 System register [A.2.8.49 TRCACATR<n>, Address Comparator Access Type Register <n> , n = 0 - 7](#) on page 1330 bits [63:0].

Attributes

Width

64

Component

ETM

Register offset

$0x480 + (8 * n)$

Access type

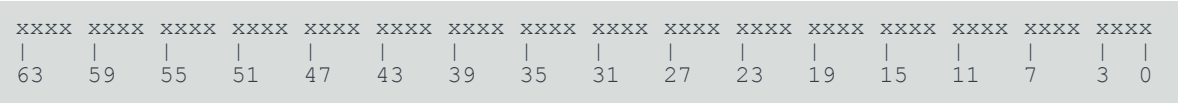
Read

R

Write

W

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-426: EXT_TRCACATR<n> bit assignments

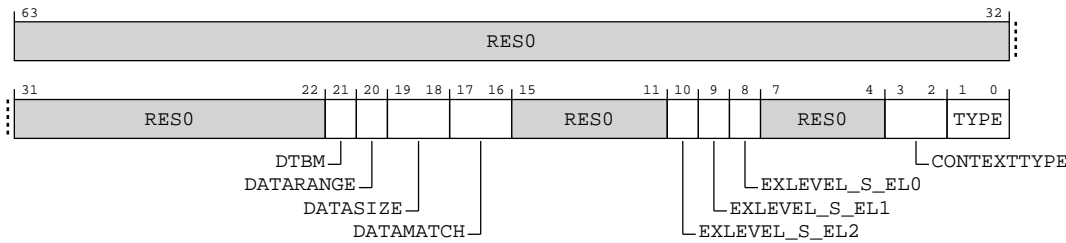


Table B-830: TRCACATR<n> bit descriptions

Bits	Name	Description	Reset
[63:22]	RES0	Reserved	RES0
[21]	DTBM	Controls whether data address comparisons use the data address [63:56] bits. 0b0 The trace unit ignores the data address [63:56] bits for data address comparisons. 0b1 The trace unit uses the data address [63:56] bits for data address comparisons.	x

Bits	Name	Description	Reset
[20]	DATARANGE	<p>When n == 0 or n == 2</p> <p>Controls whether a data value comparison uses the single address comparator or the address range comparator.</p> <p>0b0</p> <p>The trace unit uses the single address comparator for data value comparisons. The address range comparator may match at any time. The other single address comparator in the pair is not affected.</p> <p>0b1</p> <p>The trace unit uses the address range comparator for data value comparisons. The single address comparators in this pair may match at any time.</p> <p>The trace unit ignores this field when DATAMATCH==0b00.</p> <p>Otherwise</p> <p>RES0</p>	x
[19:18]	DATASIZE	<p>When n == 0 or n == 2</p> <p>Controls the width of the data value comparison.</p> <p>0b00</p> <p>Byte.</p> <p>0b01</p> <p>Halfword.</p> <p>0b10</p> <p>Word.</p> <p>0b11</p> <p>Doubleword.</p> <p>Otherwise</p> <p>RES0</p>	xx
[17:16]	DATAMATCH	<p>When n == 0 or n == 2</p> <p>Controls how the trace unit performs a data value comparison.</p> <p>0b00</p> <p>The trace unit does not perform a data value comparison.</p> <p>0b01</p> <p>The trace unit performs a data value comparison. If the data value comparator matches and the address comparator matches, the trace unit signals a match.</p> <p>0b10</p> <p>Reserved.</p> <p>0b11</p> <p>The trace unit performs a data value comparison. If the data value comparator does not match and the address comparator matches, the trace unit signals a match.</p> <p>Otherwise</p> <p>RES0</p>	xx
[15:11]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[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_ELO	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
[7:4]	RES0	Reserved	RES0
[3:2]	CONTEXTTYPE	Controls whether the address comparator is dependent on a Context Identifier Comparator, a Virtual Context Identifier Comparator, or both comparators. 0b00 The address comparator is not dependent on the Context Identifier Comparator or Virtual Context Identifier Comparator. 0b01 The address comparator is dependent on the Context Identifier Comparator. If both the Context Identifier Comparator and the address comparison match, the address comparator signals a match. 0b10 The address comparator is dependent on the Virtual Context Identifier Comparator. If both the Virtual Context Identifier Comparator and the address comparison match, the address comparator signals a match. 0b11 The address comparator is dependent on the Context Identifier Comparator and Virtual Context Identifier Comparator. If the Context Identifier Comparator, the Virtual Context Identifier Comparator and address comparison all match, the address comparator signals a match.	xx
[1:0]	TYPE	Controls what type of comparison the trace unit performs. 0b00 Instruction address. 0b01 Data load address. 0b10 Data store address. 0b11 Data load address or data store address.	xx

Access

Must be programmed if any of the following are true:

- ext-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- ext-TRCVIIECTLR.EXCLUDE[n/2] == 0b1.
- ext-TRCVIIECTLR.INCLUDE[n/2] == 0b1.
- ext-TRCVISSCTLR.START[n] == 0b1.
- ext-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.
- TRCSSCCR<>.SAC[n] == 0b1.
- ext-TRCQCTLR.RANGE[n/2] == 0b1.

Accessibility

Must be programmed if any of the following are true:

- ext-TRCBBCTLR.RANGE[n/2] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0100 and TRCRSCTLR<a>.SAC[n] == 0b1.
- TRCRSCTLR<a>.GROUP == 0b0101 and TRCRSCTLR<a>.ARC[n/2] == 0b1.
- ext-TRCVIIECTLR.EXCLUDE[n/2] == 0b1.
- ext-TRCVIIECTLR.INCLUDE[n/2] == 0b1.
- ext-TRCVISSCTLR.START[n] == 0b1.
- ext-TRCVISSCTLR.STOP[n] == 0b1.
- TRCSSCCR<>.ARC[n/2] == 0b1.
- TRCSSCCR<>.SAC[n] == 0b1.
- ext-TRCQCTLR.RANGE[n/2] == 0b1.

Component	Offset	Range
ETM	0x480 + (8 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.50 TRCDVCVR<n>, Data Value Comparator Value Register <n> , n = 0 - 1

Contains a data value.

Configurations

External register TRCDVCVR<n> bits [63:0] are architecturally mapped to AArch64 System register A.2.8.46 TRCDVCVR<n>, Data Value Comparator Value Register <n> , n = 0 - 1 on page 1323 bits [63:0].

Attributes

Width

64

Component

ETM

Register offset

0x500 + (16 * n)

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-427: EXT_TRCDVCVR<n> bit assignments

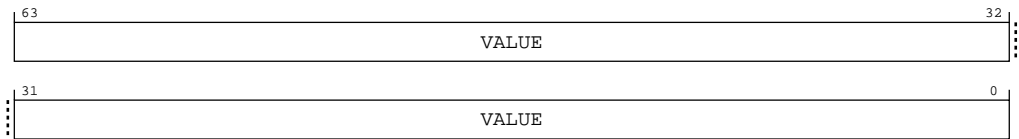


Table B-832: TRCDVCVR<n> bit descriptions

Bits	Name	Description	Reset
[63:0]	VALUE	Data Value. The data value comparators can support implementations that use multiple data widths. When the trace unit compares the VALUE field with a data value that has a width less than this field, then software must also write the comparison data value to both the upper bits and the lower bits of the VALUE field. For example, in a system that supports both 32-bit and 64-bit data widths, then software must set VALUE[63:32]=VALUE[31:0] if the trace unit is to compare a 32-bit data value.	64 {x}

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

Component	Offset	Range
ETM	0x500 + (16 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.51 TRCDVCMR<n>, Data Value Comparator Mask Register <n> , n = 0 - 1

Contains a data mask value.

Configurations

External register TRCDVCMR<n> bits [63:0] are architecturally mapped to AArch64 System register [A.2.8.47 TRCDVCMR<n>, Data Value Comparator Mask Register <n> , n = 0 - 1](#) on page 1325 bits [63:0].

Attributes

Width

64

Component

ETM

Register offset

0x580 + (16 * n)

Access type

Read

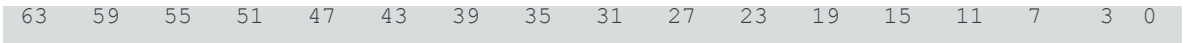
R

Write

W

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-428: EXT_TRCDVCMR<n> bit assignments

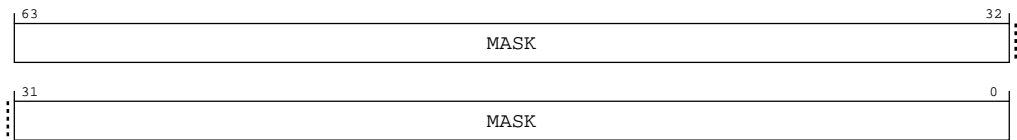


Table B-834: TRCDVCMR<n> bit descriptions

Bits	Name	Description	Reset
[63:0]	MASK	<p>Data mask value.</p> <p>If a bit is set to 1 in the mask then the comparator ignores that bit number for a data value comparison. Software must ensure that the relevant bit in ext-TRCDVCVR<n> is programmed to 0, otherwise the comparator might fail to match.</p> <p>The data value comparators can support implementations that use multiple data widths. When the trace unit compares the ext-TRCDVCVR<n>.VALUE field with a data value that has a width less than this field, then software must also write the data mask value to both the upper bits and the lower bits of the MASK field. For example, in a system that supports both 32-bit and 64-bit data widths, then software must set MASK[63:32]==MASK[31:0] if the trace unit is to compare a 32-bit data value.</p>	64 {x}

Accessibility

Might ignore writes when the trace unit is enabled or not idle.

Component	Offset	Range
ETM	0x580 + (16 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.52 TRCCIDCVR<n>, Context Identifier Comparator Value Registers <n> , n = 0

Contains a Context identifier value.

Configurations

External register TRCCIDCVR<n> bits [63:0] are architecturally mapped to AArch64 System register [A.2.8.52 TRCCIDCVR<n>, Context Identifier Comparator Value Registers <n> , n = 0](#) on page 1340 bits [63:0].

Attributes

Width

64

Component

ETM

Register offset

0x600 + (8 * n)

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-429: EXT_TRCCIDCVR<n> bit assignments

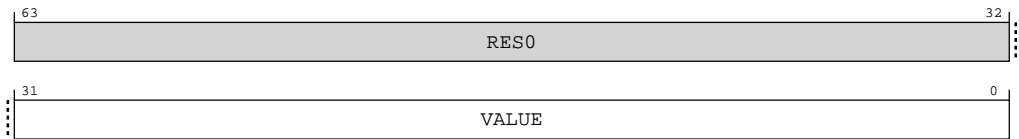


Table B-836: TRCCIDCVR<n> bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	VALUE	Context identifier value. After a PE Reset, the trace unit assumes that the Context identifier is zero until the PE updates the Context identifier.	32 {x}

Access

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0110 and TRCRSCTLR<a>.CID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b01 or 0b11 and TRCACATR<a>.CONTEXT == n.

Accessibility

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0110 and TRCRSCTLR<a>.CID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b01 or 0b11 and TRCACATR<a>.CONTEXT == n.

Component	Offset	Range
ETM	0x600 + (8 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.53 TRCVMIDCVR<n>, Virtual Context Identifier Comparator Value Register <n> , n = 0

Contains the Virtual Context Identifier Comparator value.

Configurations

External register TRCVMIDCVR<n> bits [63:0] are architecturally mapped to AArch64 System register [A.2.8.53 TRCVMIDCVR<n>, Virtual Context Identifier Comparator Value Register <n> , n = 0](#) on page 1342 bits [63:0].

Attributes**Width**

64

Component

ETM

Register offset

0x640 + (8 * n)

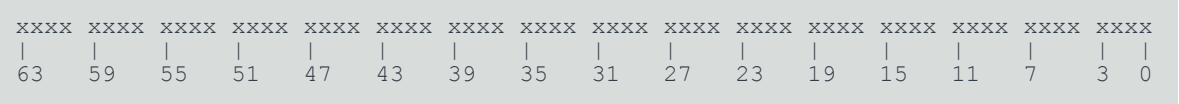
Access type

Read

R

Write
W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-430: EXT_TRCVMIDCVR<n> bit assignments

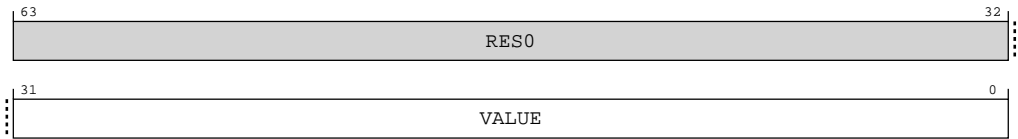


Table B-838: TRCVMIDCVR<n> bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31:0]	VALUE	Virtual context identifier value. After a PE Reset, the trace unit assumes that the Virtual context identifier is zero until the PE updates the Virtual context identifier.	32 {x}

Access

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0111 and TRCRSCTLR<a>.VMID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b10 or 0b11 and TRCACATR<a>.CONTEXT == n.

Accessibility

Must be programmed if any of the following are true:

- TRCRSCTLR<a>.GROUP == 0b0111 and TRCRSCTLR<a>.VMID[n] == 0b1.
- TRCACATR<a>.CONTEXTTYPE == 0b10 or 0b11 and TRCACATR<a>.CONTEXT == n.

Component	Offset	Range
ETM	0x640 + (8 * n)	None

This interface is accessible as follows:

RW

B.2.2.7.54 TRCCIDCCTLR0, Context Identifier Comparator Control Register 0

Contains Context identifier mask values for the ext-TRCCIDCVR<n> registers, for n = 0 to 3.

Configurations

External register TRCCIDCCTLR0 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.50 TRCCIDCCTLR0, Context Identifier Comparator Control Register 0](#) on page 1335 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x680

Access type

Read

R

Write

W

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-431: EXT_TRCCIDCCTLR0 bit assignments

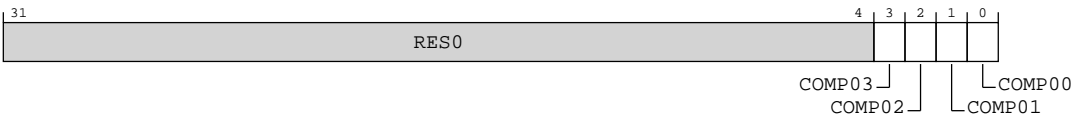


Table B-840: TRCCIDCTLR0 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	COMP0<m>, bit[m], where m = 3 to 0	<p>TRCCIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0.</p> <p>0b0</p> <p>The trace unit includes TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.</p> <p>0b1</p> <p>The trace unit ignores TRCCIDCVR0[(m×8+7):(m×8)] when it performs the Context identifier comparison.</p>	xxxx

Access

If software uses the ext-TRCCIDCVR<n> registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in ext-TRCCIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in ext-TRCCIDCVR<n> is not 0x00, the Context Identifier Comparator will not match.

Accessibility

If software uses the ext-TRCCIDCVR<n> registers, for n = 0 to 3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in ext-TRCCIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in ext-TRCCIDCVR<n> is not 0x00, the Context Identifier Comparator will not match.

Component	Offset	Range
ETM	0x680	None

This interface is accessible as follows:

RW

B.2.2.7.55 TRCVMIDCCTLR0, Virtual Context Identifier Comparator Control Register 0

Virtual Context Identifier Comparator mask values for the ext-TRCVMIDCVR<n> registers, where n=0-3.

Configurations

External register TRCVMIDCCTLR0 bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.51 TRCVMIDCCTLR0, Virtual Context Identifier Comparator Control Register 0](#) on page 1337 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0x688

Access type

Read

R

Write

W

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-432: EXT_TRCVMIDCCTLR0 bit assignments

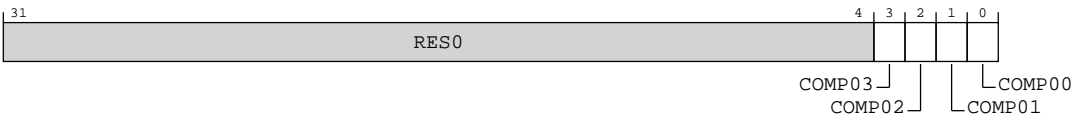


Table B-842: TRCVMIDCCTLR0 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	COMP0<m>, bit[m], where m = 3 to 0	<p>TRCVMIDCVR0 mask control. Specifies the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0.</p> <p>0b0</p> <p>The trace unit includes TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.</p> <p>0b1</p> <p>The trace unit ignores TRCVMIDCVR0[(m×8+7):(m×8)] when it performs the Virtual context identifier comparison.</p>	xxxx

Access

If software uses the ext-TRCVMIDCVR<n> registers, where n=0-3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in ext-TRCVMIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in ext-TRCVMIDCVR<n> is not 0x00, the Virtual Context Identifier Comparator will not match.

Accessibility

If software uses the ext-TRCVMIDCVR<n> registers, where n=0-3, then it must program this register.

If software sets a mask bit to 0b1 then it must program the relevant byte in ext-TRCVMIDCVR<n> to 0x00.

If any bit is 0b1 and the relevant byte in ext-TRCVMIDCVR<n> is not 0x00, the Virtual Context Identifier Comparator will not match.

Component	Offset	Range
ETM	0x688	None

This interface is accessible as follows:

RW

B.2.2.7.56 TRCITCTRL, Integration Mode Control Register

No functional/integration mode switching implemented in the trace unit.

Configurations

This register is available in all configurations.

Attributes

Width
32

Component
ETM

Register offset
0xF00

Access type
RW

Reset value

xxxx

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-433: EXT_TRCITCTRL bit assignments

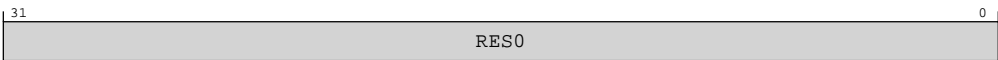


Table B-844: TRCITCTRL bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0xF00	None

This interface is accessible as follows:

RW

B.2.2.7.57 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.

Configurations

External register TRCCLAIMSET bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.55 TRCCLAIMSET, Claim Tag Set Register](#) on page 1345 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0xFA0

Access type

RAOW1S

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1111
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-434: EXT_TRCCLAIMSET bit assignments

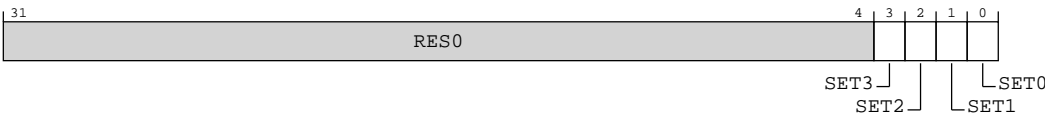


Table B-846: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[3:0]	SET<m>, bit[m], where m = 3 to 0	<div>Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1.</div> <div>0b0<div>On a read: Claim Tag bit m is not implemented.</div><div>On a write: Ignored.</div></div> <div>0b1<div>On a read: Claim Tag bit m is implemented.</div><div>On a write: Set Claim Tag bit m to 0b1.</div></div>	0b1111

Accessibility

Component	Offset	Range
ETM	0xFA0	None

This interface is accessible as follows:

RW

B.2.2.7.58 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with ext-TRCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

Configurations

External register TRCCLAIMCLR bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.56 TRCCLAIMCLR, Claim Tag Clear Register](#) on page 1347 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0xFA4

Access type

RW1C

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-435: EXT_TRCCLAIMCLR bit assignments

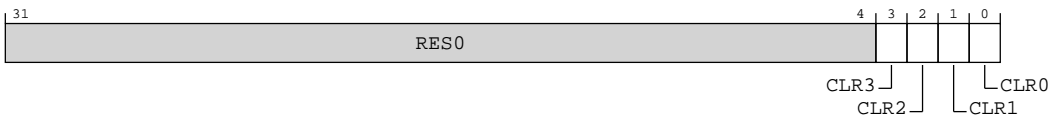


Table B-848: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	CLR<m>, bit[m], where m = 3 to 0	<p>Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0.</p> <p>0b0</p> <p>On a read: Claim Tag bit m is not set.</p> <p>On a write: Ignored.</p> <p>0b1</p> <p>On a read: Claim Tag bit m is set.</p> <p>On a write: Clear Claim tag bit m to 0b0.</p> <p>The number of Claim Tag bits implemented is indicated in ext-TRCCLAIMSET.</p>	xxxx

Accessibility

Component	Offset	Range
ETM	0xFA4	None

This interface is accessible as follows:

RW

B.2.2.7.59 TRCDEVAFF, Device Affinity Register

Reads the same value as the AArch64-MPIDR_EL1 register for the PE this trace unit has affinity with.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

ETM

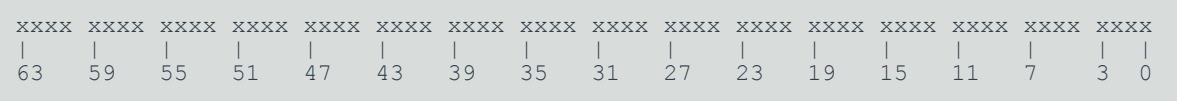
Register offset

0xFA8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-436: EXT_TRCDEVAFF bit assignments

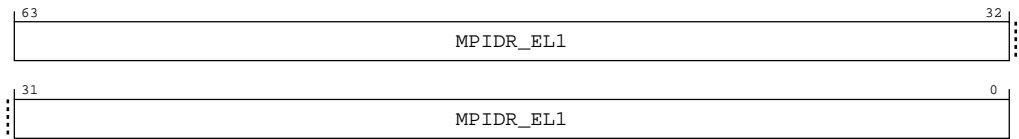


Table B-850: TRCDEVAFF bit descriptions

Bits	Name	Description	Reset
[63:0]	MPIDR_EL1	Read-only copy of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	64 { x }

Accessibility

Component	Offset	Range
ETM	0xFA8	None

This interface is accessible as follows:

RO

B.2.2.7.60 TRCLAR, Lock Access Register

Legacy Software Lock mechanism.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFB0

Access type

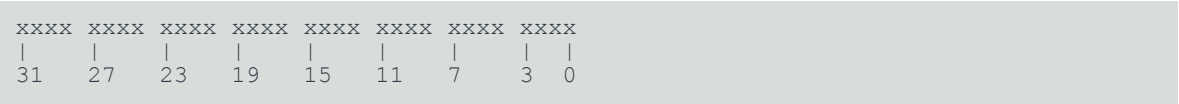
Read

RESERVED

Write

W

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-437: EXT_TRCLAR bit assignments

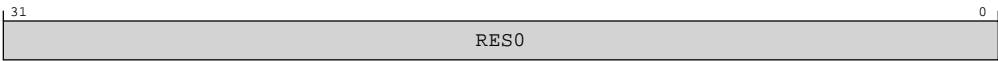


Table B-852: TRCLAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Software Lock not implemented, writes to TRCLAR are ignored.

Component	Offset	Range
ETM	0xFB0	None

This interface is accessible as follows:

WO

B.2.2.7.61 TRCLSR, Lock Status Register

Legacy Software Lock mechanism.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFB4

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxx0
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-438: EXT_TRCLSR bit assignments

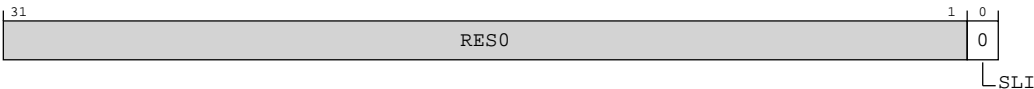


Table B-854: TRCLSR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0
[0]	SLI	Indicates whether the Software Lock is implemented. 0b0 Software Lock is not implemented. Writes to the ext-TRCLAR are ignored.	0b0

Accessibility

Component	Offset	Range
ETM	0xFB4	None

This interface is accessible as follows:

RO

B.2.2.7.62 TRCAUTHSTATUS, Authentication Status Register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for debug.

Configurations

External register TRCAUTHSTATUS bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.57 TRCAUTHSTATUS, Authentication Status Register](#) on page 1349 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0xFB8

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xx00	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-439: EXT_TRCAUTHSTATUS bit assignments

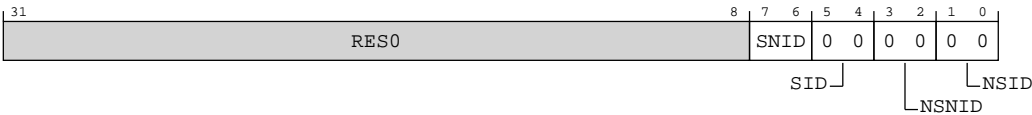


Table B-856: TRCAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:6]	SNID	Secure Non-invasive Debug. Indicates whether Secure non-invasive debug features are implemented and enabled. 0b10 Secure Non-invasive Debug implemented and disabled. 0b11 Secure Non-invasive Debug implemented and enabled.	xx
[5:4]	SID	Secure Invasive Debug. Indicates whether Secure invasive debug features are implemented and enabled. 0b00 Secure invasive debug features not implemented.	0b00
[3:2]	NSNID	Non-secure Non-invasive Debug. Indicates whether Non-secure non-invasive debug features are implemented and enabled. 0b00 Non-secure non-invasive debug features not implemented.	0b00
[1:0]	NSID	Non-secure Invasive Debug. Indicates whether Non-secure invasive debug features are implemented and enabled. 0b00 Non-secure invasive debug features not implemented.	0b00

Accessibility

Component	Offset	Range
ETM	0xFB8	None

This interface is accessible as follows:

RO

B.2.2.7.63 TRCDEVARCH, Device Architecture Register

Provides discovery information for the component.

Configurations

External register TRCDEVARCH bits [31:0] are architecturally mapped to AArch64 System register A.2.8.58 TRCDEVARCH, Device Architecture Register on page 1351 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0xFBC

Access type

RO

Reset value

0100 0111 0111 0101 0100 1010 0001 0011

Bit descriptions

Figure B-440: EXT_TRCDEVARCH bit assignments

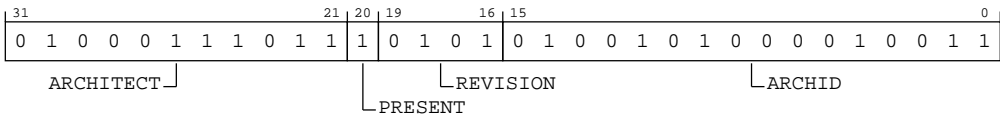


Table B-858: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. Defines the architect of the component. 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. 0b0101 ETMv4.5.	0b0101
[15:0]	ARCHID	Architecture ID. 0b0100101000010011 Arm PE Trace architecture ETMv4.	0xA13

Accessibility

Component	Offset	Range
ETM	0xFBC	None

This interface is accessible as follows:

RO

B.2.2.7.64 TRCDEVID, Device Configuration Register

Provides discovery information for the component.

Configurations

External register TRCDEVID bits [31:0] are architecturally mapped to AArch64 System register [A.2.8.54 TRCDEVID, Device Configuration Register](#) on page 1344 bits [31:0].

Attributes

Width

32

Component

ETM

Register offset

0xFC8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-441: EXT_TRCDEVID bit assignments

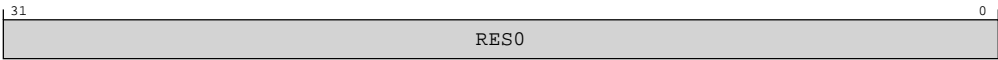


Table B-860: TRCDEVID bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0xFC8	None

This interface is accessible as follows:

RO

B.2.2.7.65 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognised, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

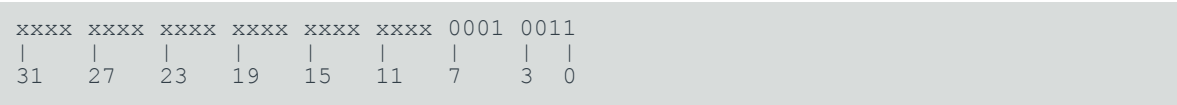
Register offset

0xFCC

Access type

RO

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-442: EXT_TRCDEVTYPE bit assignments

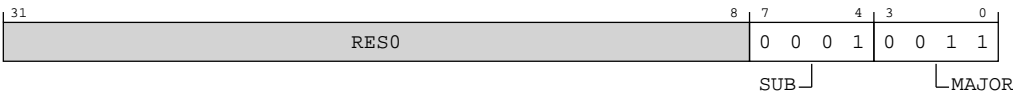


Table B-862: TRCDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	SUB	Component sub-type. 0b0001 Trace source associated with a PE.	0b0001
[3:0]	MAJOR	Component major type. 0b0011 Trace source.	0b0011

Accessibility

Component	Offset	Range
ETM	0xFCC	None

This interface is accessible as follows:

RO

B.2.2.7.66 TRCPIDR4, Peripheral Identification Register 4

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

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-443: EXT_TRCPIDR4 bit assignments

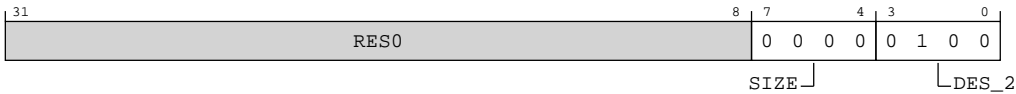


Table B-864: TRCPIDR4 bit descriptions

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
ETM	0xFD0	None

This interface is accessible as follows:

RO

B.2.2.7.67 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

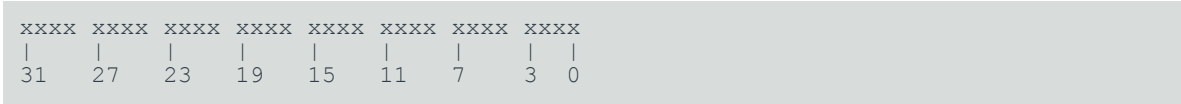
Register offset

0xFD4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-444: EXT_TRCPIDR5 bit assignments

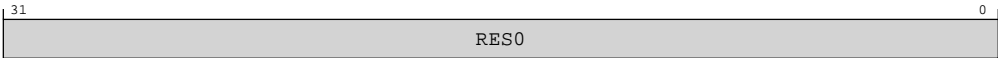


Table B-866: TRCPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0xFD4	None

This interface is accessible as follows:

RO

B.2.2.7.68 TRCPIDR6, Peripheral Identification Register 6

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFD8

Access type
RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-445: EXT_TRCPIDR6 bit assignments



Table B-868: TRCPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0xFD8	None

This interface is accessible as follows:

RO

B.2.2.7.69 TRCPIDR7, Peripheral Identification Register 7

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFDC

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-446: EXT_TRCPIDR7 bit assignments



Table B-870: TRCPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Range
ETM	0xFDC	None

This interface is accessible as follows:

RO

B.2.2.7.70 TRCPIDR0, Peripheral Identification Register 0

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

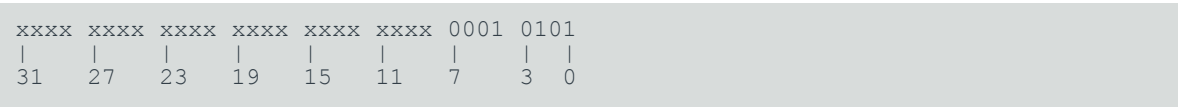
Register offset

0xFE0

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-447: EXT_TRCPIDR0 bit assignments

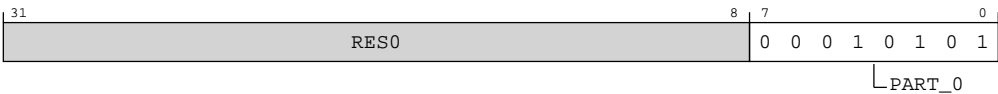


Table B-872: TRCPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b00010101 Cortex-R82 trace unit. Bits [7:0] of part number 0xD15.	0x15

Accessibility

Component	Offset	Range
ETM	0xFE0	None

This interface is accessible as follows:

RO

B.2.2.7.71 TRCPIDR1, Peripheral Identification Register 1

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

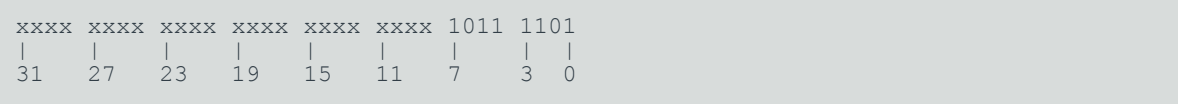
Register offset

0xFE4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-448: EXT_TRCPIDR1 bit assignments

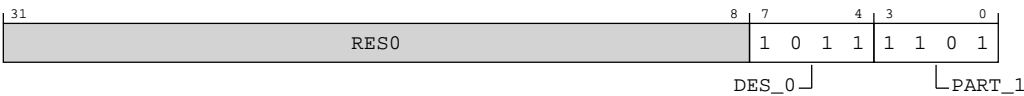


Table B-874: TRCPIDR1 bit descriptions

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 Cortex-R82 trace unit. Bits [11:8] of part number 0xD14.	0b1101

Accessibility

Component	Offset	Range
ETM	0xFE4	None

This interface is accessible as follows:

RO

B.2.2.7.72 TRCPIDR2, Peripheral Identification Register 2

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFE8

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0111	1011
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-449: EXT_TRCPIDR2 bit assignments

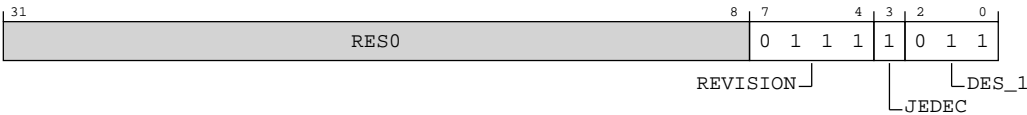


Table B-876: TRCPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Range
ETM	0xFE8	None

This interface is accessible as follows:

RO

B.2.2.7.73 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFEC

Access type

RO

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0000	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-450: EXT_TRCPIDR3 bit assignments



Table B-878: TRCPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Minor errata fixes. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer Modified. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Range
ETM	0xFEC	None

This interface is accessible as follows:

RO

B.2.2.7.74 TRCCIDR0, Component Identification Register 0

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

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



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-451: EXT_TRCCIDR0 bit assignments

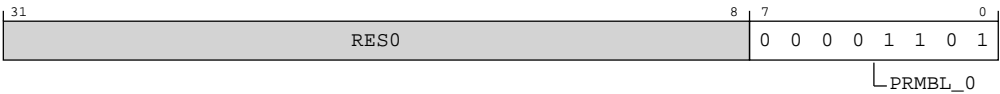


Table B-880: TRCCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble. 0b00001101 CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Range
ETM	0xFF0	None

This interface is accessible as follows:

RO

B.2.2.7.75 TRCCIDR1, Component Identification Register 1

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFF4

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-452: EXT_TRCCIDR1 bit assignments



Table B-882: TRCCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	CLASS	Component class. 0b1001 CoreSight debug component.	0b1001
[3:0]	PRMBL_1	Preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Range
ETM	0xFF4	None

This interface is accessible as follows:

RO

B.2.2.7.76 TRCCIDR2, Component Identification Register 2

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

Register offset

0xFF8

Access type

RO

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-453: EXT_TRCCIDR2 bit assignments

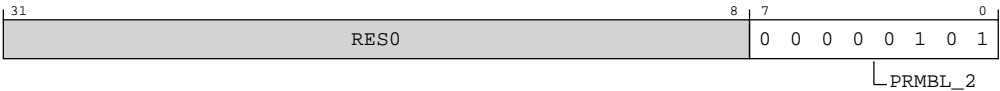


Table B-884: TRCCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble. 0b00000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Range
ETM	0xFF8	None

This interface is accessible as follows:

RO

B.2.2.7.77 TRCCIDR3, Component Identification Register 3

Provides discovery information for the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Component

ETM

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-454: EXT_TRCCIDR3 bit assignments



Table B-886: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble. 0b10110001 CoreSight component identification preamble.	0xB1

Accessibility

Component	Offset	Range
ETM	0xFFC	None

This interface is accessible as follows:

RO

B.2.2.8 External PMU register description

This section includes the register descriptions for all memory-mapped *Performance Monitoring Unit* (PMU) registers that are accessed for each core.

B.2.2.8.1 PMEVCNTR<n>_ELO, Performance Monitors Event Count Registers, n = 0 - 5

Holds event counter <n>, which counts events, where <n> is 0 to 30.

Configurations

PMEVCNTR<n>_ELO is in the Core power domain.

External register PMEVCNTR<n>_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.32 PMEVCNTR<n>_ELO, Performance Monitors Event Count Registers, n = 0 - 5](#) on page 806 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0x000 + (8 * n)

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-455: PMU_PMEVCNTR<n>_ELO bit assignments

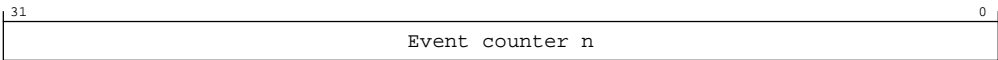


Table B-888: PMEVCNTR<n>_ELO bit descriptions

Bits	Name	Description	Reset
[31:0]	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.	32 { x }

Accessibility

Component	Offset	Instance	Range
PMU	0x000 + (8 * n)	PMEVCNTR<n>_ELO	31:0

B.2.2.8.2 PMCCNTR_ELO, Performance Monitors Cycle Counter

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. For more information, see *Time as measured by the Performance Monitors cycle counter* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

PMU.PMCCFILTR_ELO determines the modes and states in which the PMCCNTR_ELO can increment.

Configurations

PMCCNTR_ELO is in the Core power domain.

External register PMCCNTR_ELO bits [63:0] are architecturally mapped to AArch64 System register [A.2.3.27 PMCCNTR_ELO, Performance Monitors Cycle Count Register](#) on page 793 bits [63:0].

Attributes

Width

64

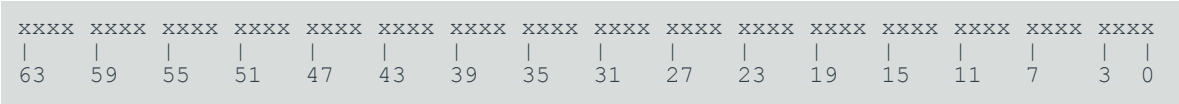
Component

PMU

Register offsets (2)

0x0F8,0x0FC

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-456: PMU_PMCNTR_ELO bit assignments

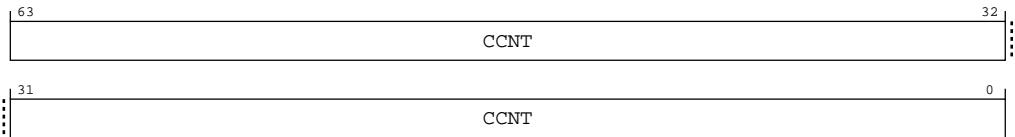


Table B-890: PMCCNTR_ELO bit descriptions

Bits	Name	Description	Reset
[63:0]	CCNT	Cycle count. Depending on the values of PMU.PMCR_ELO.{LC,D}, the cycle count increments in one of the following ways: <ul style="list-style-type: none">Every processor clock cycle.Every 64th processor clock cycle. Writing 1 to PMU.PMCR_ELO.C sets this field to 0.	64 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0x0F8	PMCCNTR_ELO	31:0

Component	Offset	Instance	Range
PMU	0x0FC	PMCCNTR_ELO	63:32

B.2.2.8.3 PMPCSR, Program Counter Sample Register

Holds a sampled instruction address value.

Configurations

This register is available in all configurations.

Attributes

Width

64

Component

PMU

Register offsets (4)

0x200,0x204,0x220,0x224

Reset value

0xxx	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-457: PMU_PMPCSR bit assignments

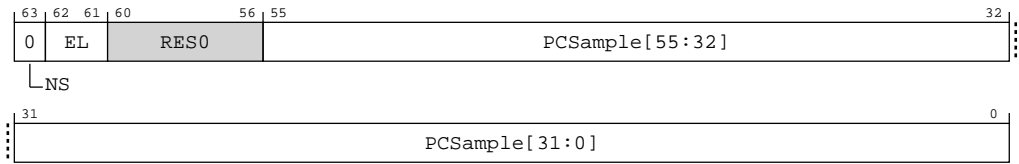


Table B-893: PMPCSR bit descriptions

Bits	Name	Description	Reset
[63]	NS	Non-secure state sample. Indicates the Security state that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample. If EL3 is not implemented, this bit indicates the Effective value of SCR.NS. 0b0 Sample is from Secure state.	0b0
[62:61]	EL	Exception level status sample. Indicates the Exception level that is associated with the most recent PMPCSR sample or, when it is read as a single atomic 64-bit read, the current PMPCSR sample. 0b00 Sample is from EL0. 0b01 Sample is from EL1. 0b10 Sample is from EL2.	xx
[60:56]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[55:32]	PCSample[55:32]	Bits[55:32] of the sampled instruction address value. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.	24 {x}
[31:0]	PCSample[31:0]	<p>Bits[31:0] of the sampled instruction address value.</p> <p>PMPCSR[31:0] reads as 0xFFFFFFFF when any of the following are true:</p> <ul style="list-style-type: none"> The PE is in Debug state. PC Sample-based profiling is prohibited. <p>If a branch instruction has retired since the PE left reset state, then the first read of PMPCSR[31:0] is permitted but not required to return 0xFFFFFFFF.</p> <p>PMPCSR[31:0] reads as an UNKNOWN value when any of the following are true:</p> <ul style="list-style-type: none"> The PE is in reset state. No branch instruction has retired since the PE left reset state, Debug state, or a state where PC Sample-based Profiling is prohibited. No branch instruction has retired since the last read of PMPCSR[31:0]. <p>For the cases where a read of PMPCSR[31:0] returns 0xFFFFFFFF or an UNKNOWN value, the read has the side-effect of setting PMPCSR[63:32], ext-PMCID1SR, ext-PMCID2SR, and ext-PMVIDSR to UNKNOWN values.</p> <p>Otherwise, a read of PMPCSR[31:0] returns bits [31:0] of the sampled instruction address value and has the side-effect of indirectly writing to PMPCSR[63:32], ext-PMCID1SR, ext-PMCID2SR, and ext-PMVIDSR. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.</p>	32 {x}

Access

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register **UNKNOWN**, see *Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN* in the [Arm® Architecture Reference Manual for A-profile architecture](#).



Note

A 32-bit access to PMPCSR[63:32] does not update the PC sample registers. Only a 64-bit access to PMPCSR[63:0] or a 32-bit access to PMPCSR[31:0] updates the PC sample registers. This includes the value a subsequent 32-bit read of PMPCSR[63:32] will return.

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register **UNKNOWN**, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.



Note

A 32-bit access to PMPCSR[63:32] does not update the PC sample registers. Only a 64-bit access to PMPCSR[63:0] or a 32-bit access to PMPCSR[31:0] updates the PC sample registers. This includes the value a subsequent 32-bit read of PMPCSR[63:32] will return.

Component	Offset	Instance	Range
PMU	0x200	PMPCSR	31:0

Component	Offset	Instance	Range
PMU	0x204	PMPCSR	63:32

Component	Offset	Instance	Range
PMU	0x220	PMPCSR	31:0

Component	Offset	Instance	Range
PMU	0x224	PMPCSR	63:32

B.2.2.8.4 PMCID1SR, CONTEXTIDR_EL1 Sample Register

Contains the sampled value of AArch64-CONTEXTIDR_EL1, captured on reading PMU.PMPCSR[31:0].

Configurations

This register is available in all configurations.

Attributes

Width

32

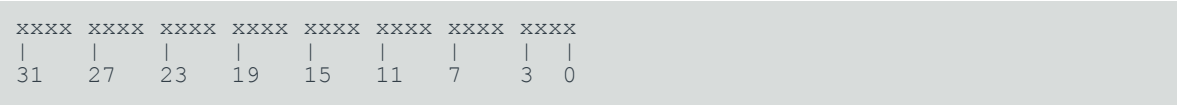
Component

PMU

Register offsets (2)

0x208,0x228

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-458: PMU_PMCID1SR bit assignments



Table B-898: PMCID1SR bit descriptions

Bits	Name	Description	Reset
[31:0]	CONTEXTIDR_EL1	<div>Context ID. The value of CONTEXTIDR that is associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample is generated:<ul style="list-style-type: none">If EL1 is using AArch64, then the Context ID is sampled from AArch64-CONTEXTIDR_EL1.Because the value written to this register is an indirect read of CONTEXTIDR, it is CONSTRAINED UNPREDICTABLE whether this register is set to the original or new value if PMU.PMPCSR samples:<ul style="list-style-type: none">An instruction that writes to CONTEXTIDR.The next Context synchronization event.Any instruction executed between these two instructions.</div>	32 {x}

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Component	Offset	Instance	Range
PMU	0x208	PMCID1SR	None

Component	Offset	Instance	Range
PMU	0x228	PMCID1SR	None

B.2.2.8.5 PMVIDSR, VMID Sample Register

Contains the sampled VMID value that is captured on reading PMU.PMPCSR[31:0].

Configurations

This register is available in all configurations.

Attributes

Width

32

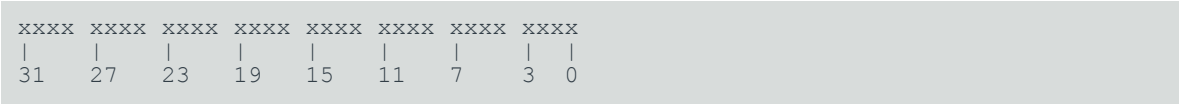
Component

PMU

Register offset

0x20C

Reset value



**Note**

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-459: PMU_PMVIDSR bit assignments



Table B-901: PMVIDSR bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	VMID	<p>VMID sample. The VMID associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample was generated:</p> <ul style="list-style-type: none"> This field is set to an UNKNOWN value if any of the following apply: <ul style="list-style-type: none"> The PE is executing at EL2. Otherwise: <ul style="list-style-type: none"> If EL2 is using AArch64, this field is set to AArch64-VSCTLR_EL2.VMID. <p>Because the value written to PMVIDSR is an indirect read of the VMID value, it is CONSTRAINED UNPREDICTABLE whether PMVIDSR is set to the original or new value if PMU.PMPCSR samples:</p> <ul style="list-style-type: none"> An instruction that writes to the VMID value. The next Context synchronization event. Any instruction executed between these two instructions. 	8 {x}

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Component	Offset	Instance	Range
PMU	0x20C	PMVIDSR	None

B.2.2.8.6 PMCID2SR, CONTEXTIDR_EL2 Sample Register

Contains the sampled value of AArch64-CONTEXTIDR_EL2, captured on reading PMU.PMPCSR[31:0].

Configurations

This register is available in all configurations.

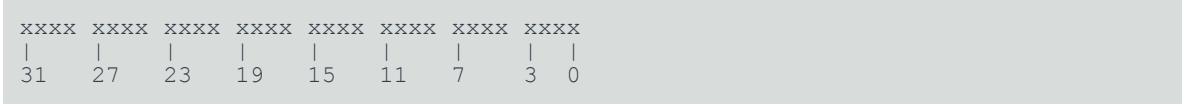
Attributes

Width
32

Component
PMU

Register offset
0x22C

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-460: PMU_PMCID2SR bit assignments



Table B-903: PMCID2SR bit descriptions

Bits	Name	Description	Reset
[31:0]	CONTEXTIDR_EL2	<p>Context ID. The value of AArch64-CONTEXTIDR_EL2 that is associated with the most recent PMU.PMPCSR sample. When the most recent PMU.PMPCSR sample is generated:</p> <ul style="list-style-type: none">If the PE is not executing at EL3, EL2 is using AArch64, and EL2 is enabled in the current Security state, then this field is set to the Context ID sampled from AArch64-CONTEXTIDR_EL2.Otherwise, this field is set to an UNKNOWN value. <p>Because the value written to this field is an indirect read of AArch64-CONTEXTIDR_EL2, it is CONSTRAINED UNPREDICTABLE whether this field is set to the original or new value if PMU.PMPCSR samples:</p> <ul style="list-style-type: none">An instruction that writes to AArch64-CONTEXTIDR_EL2.The next Context synchronization event.Any instruction executed between these two instructions.	32 {x}

Accessibility

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN'.

Component	Offset	Instance	Range
PMU	0x22C	PMCID2SR	None

B.2.2.8.7 [PMEVTYPER<n>_ELO](#), Performance Monitors Event Type Registers, n = 0 - 5

Configures event counter <n>, where <n> is 0 to 30.

Configurations

PMEVTYPER<n>_ELO is in the Core power domain.

If event counter n is not implemented:

- When `IsCorePowered()` && `!DoubleLockStatus()` && `!OSLockStatus()` && `AllowExternalPMUAccess()`, accesses are RES0.
- Otherwise, it is CONSTRAINED UNPREDICTABLE whether accesses to this register are RES0 or generate an error response.

External register [PMEVTYPER<n>_ELO](#) bits [63:0] are architecturally mapped to AArch64 System register [A.2.3.33 PMEVTYPER<n>_ELO](#), Performance Monitors Event Type Registers, n = 0 - 5 on page 809 bits [63:0].

Attributes

Width

64

Component

PMU

Register offset

0x400 + (4 * n)

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 B-461: PMU_PMEVTYPER<n>_ELO bit assignments

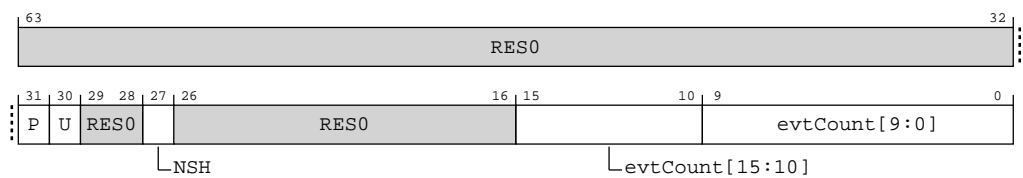


Table B-905: PMEVTYPER<n>_ELO bit descriptions

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	ELO filtering. Controls counting events in ELO. 0b0 This field has no effect on filtering of events. 0b1 Events in ELO are not counted.	x
[29:28]	RES0	Reserved	RES0
[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
[26:16]	RES0	Reserved	RES0
[15:10]	evtCount[15:10]	Extension to evtCount[9:0]. For more information, see evtCount[9:0].	6 { x }

Bits	Name	Description	Reset
[9:0]	evtCount[9:0]	<p>Event to count. The event number of the event that is counted by event counter PMU.PMEVCNTR<n>_ELO.</p> <p>Software must program this field with an event that is supported by the PE being programmed.</p> <p>The ranges of event numbers allocated to each type of event are shown in <i>Allocation of the PMU event number space</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p> <p>If FEAT_PMUv3p8 is implemented and PMEVTYPER<n>_ELO.evtCount is programmed to an event that is reserved or not supported by the PE, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_ELO.evtCount field is the value written to the field.</p> <p>Note: Arm recommends this behavior for all implementations of FEAT_PMUv3.</p> <p>Otherwise, if PMEVTYPER<n>_ELO.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:</p> <ul style="list-style-type: none"> For the range 0x0000 to 0x003F, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_ELO.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>_ELO.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>_ELO.evtCount field is UNKNOWN. <p>Note: UNPREDICTABLE means the event must not expose privileged information.</p>	10 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0x400 + (4 * n)	PMEVTYPER<n>_ELO	31:0

B.2.2.8.8 PMCCFILTR_ELO, Performance Monitors Cycle Counter Filter Register

Determines the modes in which the Cycle Counter, PMU.PMCCNTR_ELO, increments.

Configurations

PMCCFILTR_ELO is in the Core power domain.

On a Warm or Cold reset, RW fields in this register reset to:

- Architecturally UNKNOWN values if the reset is to an Exception level that is using AArch64.

The register is not affected by an External debug reset.

External register PMCCFILTR_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.34 PMCCFILTR_ELO, Performance Monitors Cycle Count Filter Register](#) on page 812 bits [31:0].

Attributes

Width

64

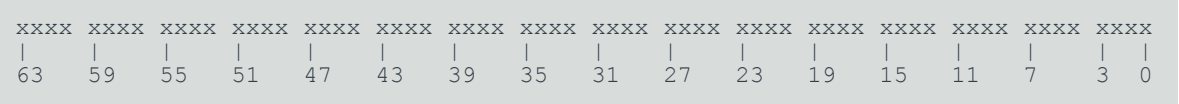
Component

PMU

Register offset

0x47C

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-462: PMU_PMCCFILTR_EL0 bit assignments

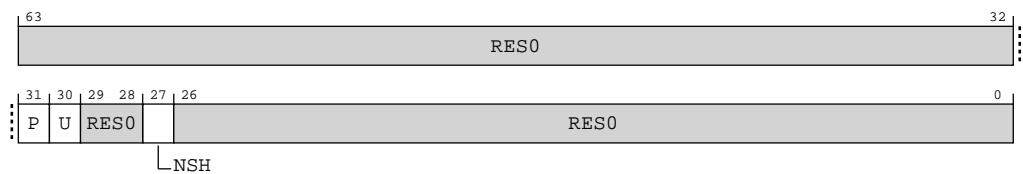


Table B-907: PMCCFILTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	RES0
[31]	P	EL1 filtering. Controls counting cycles in EL1. 0b0 This field has no effect on filtering of cycles. 0b1 Cycles in EL1 are not counted.	x
[30]	U	EL0 filtering. Controls counting cycles in EL0. 0b0 This field has no effect on filtering of cycles. 0b1 Cycles in EL0 are not counted.	x
[29:28]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[27]	NSH	EL2 filtering. Controls counting cycles in EL2. 0b0 Cycles in EL2 are not counted. 0b1 This field has no effect on filtering of cycles.	x
[26:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
PMU	0x47C	PMCCFILTR_ELO	31:0

B.2.2.8.9 [PMCNTENSET_ELO, Performance Monitors Count Enable Set Register](#)

Enables the Cycle Count Register, PMU.PMCCNTR_ELO, and any implemented event counters PMU.PMEVCNTR<n>_ELO. Reading this register shows which counters are enabled.

Configurations

PMCNTENSET_ELO is in the Core power domain.

External register PMCNTENSET_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.20 PMCNTENSET_ELO, Performance Monitors Count Enable Set Register](#) on page 763 bits [31:0].

External register PMCNTENSET_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.21 PMCNTENCLR_ELO, Performance Monitors Count Enable Clear Register](#) on page 766 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xC00

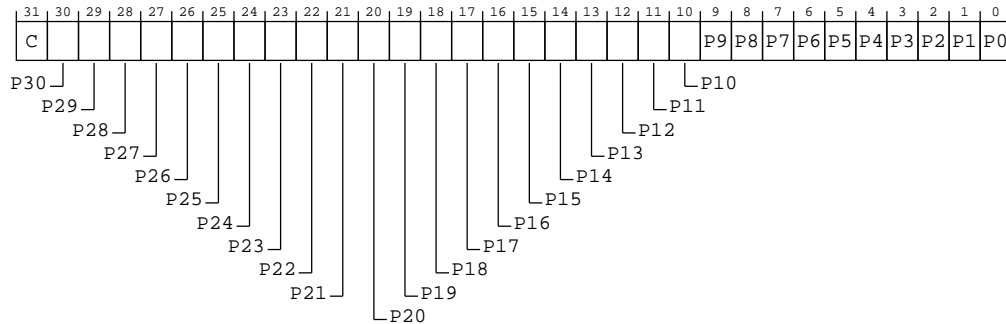
Reset value

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-463: PMU_PMCNTENSET_ELO bit assignments**Table B-909: PMCNTENSET_ELO bit descriptions**

Bits	Name	Description	Reset
[31]	C	<p>PMU.PMCCNTR_ELO enable. On writes, allows software to enable PMU.PMCCNTR_ELO. On reads, returns the PMU.PMCCNTR_ELO enable status.</p> <p>0b0</p> <p>PMU.PMCCNTR_ELO disabled.</p> <p>0b1</p> <p>PMU.PMCCNTR_ELO enabled.</p> <p>Access to this field is: W1S</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>PMEVCNTR<m>_ELO enable. On writes, allows software to enable PMEVCNTR<m>_ELO. On reads, returns the PMEVCNTR<m>_ELO enable status.</p> <p>0b0</p> <p>PMEVCNTR<m>_ELO disabled.</p> <p>0b1</p> <p>PMEVCNTR<m>_ELO enabled.</p> <p>When m >= 6</p> <p>Access to this field is: RAZ/WI</p> <p>Otherwise</p> <p>Access to this field is: W1S</p>	31 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xC00	PMCNTENSET_ELO	31:0

B.2.2.8.10 [PMCNTENCLR_ELO, Performance Monitors Count Enable Clear Register](#)

Disables the Cycle Count Register, PMU.PMCCNTR_ELO, and any implemented event counters AArch32-PMEVCNTR<n>. Reading this register shows which counters are enabled.

Configurations

PMCNTENCLR_ELO is in the Core power domain.

External register PMCNTENCLR_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.21 PMCNTENCLR_ELO, Performance Monitors Count Enable Clear Register](#) on page 766 bits [31:0].

External register PMCNTENCLR_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.20 PMCNTENSET_ELO, Performance Monitors Count Enable Set Register](#) on page 763 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xC20

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-464: PMU_PMCNTENCLR_ELO bit assignments

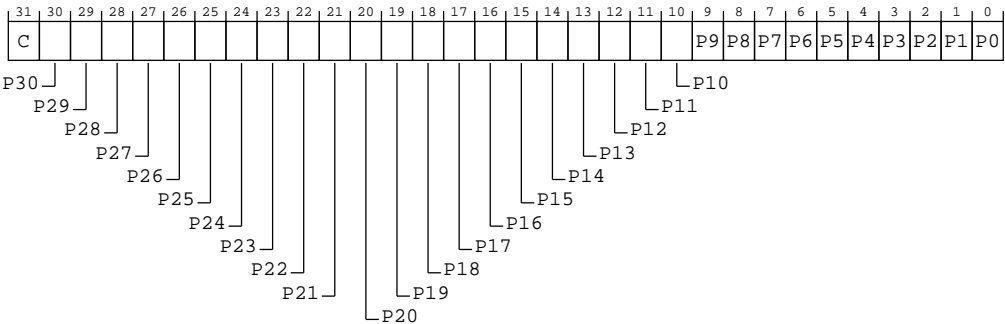


Table B-911: PMCNTENCLR_ELO bit descriptions

Bits	Name	Description	Reset
[31]	C	PMU.PMCCNTR_ELO disable. On writes, allows software to disable PMU.PMCCNTR_ELO. On reads, returns the PMU.PMCCNTR_ELO enable status. 0b0 PMU.PMCCNTR_ELO disabled. 0b1 PMU.PMCCNTR_ELO enabled. Access to this field is: W1C	x
[30:0]	P<m>, bit[m], where m = 30 to 0	PMEVCNTR<m>_ELO disable. On writes, allows software to disable PMEVCNTR<m>_ELO. On reads, returns the PMEVCNTR<m>_ELO enable status. 0b0 PMEVCNTR<m>_ELO disabled. 0b1 PMEVCNTR<m>_ELO enabled. When m >= 6 Access to this field is: RAZ/WI Otherwise Access to this field is: W1C	31 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xC20	PMCNTENCLR_ELO	31:0

B.2.2.8.11 PMINTENSET_EL1, Performance Monitors Interrupt Enable Set Register

Enables the generation of interrupt requests on overflows from the Cycle Count Register, PMU.PMCCNTR_EL0, and the event counters PMU.PMEVCNTR<n>_EL0. Reading the register shows which overflow interrupt requests are enabled.

Configurations

PMINTENSET_EL1 is in the Core power domain.

External register PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register A.2.3.1 PMINTENSET_EL1, Performance Monitors Interrupt Enable Set Register on page 710 bits [31:0].

External register PMINTENSET_EL1 bits [31:0] are architecturally mapped to AArch64 System register A.2.3.2 PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear Register on page 713 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xC40

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-465: PMU_PMINTENSET_EL1 bit assignments

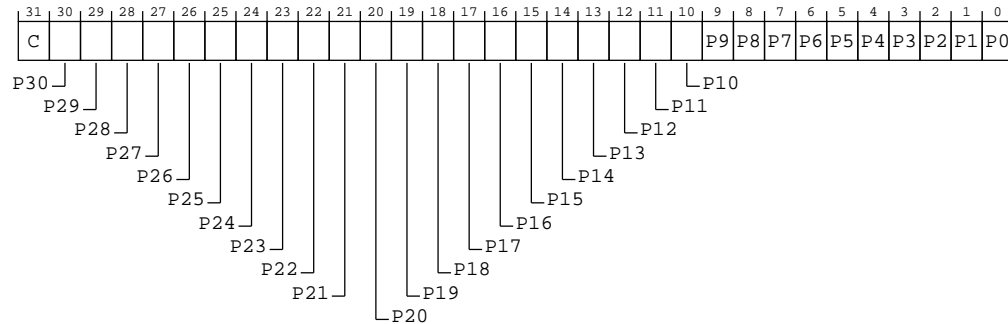


Table B-913: PMINTENSET_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	<p>Interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO enable. On writes, allows software to enable the interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO. On reads, returns the interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO enable status.</p> <p>0b0</p> <p>Interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO disabled.</p> <p>0b1</p> <p>Interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO enabled.</p> <p>Access to this field is: W1S</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO enable. On writes, allows software to enable the interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO. On reads, returns the interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO enable status.</p> <p>0b0</p> <p>Interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO disabled.</p> <p>0b1</p> <p>Interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO enabled.</p> <p>When m >= 6</p> <p>Access to this field is: RAZ/WI</p> <p>Otherwise</p> <p>Access to this field is: W1S</p>	31 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xC40	PMINTENSET_EL1	31:0

B.2.2.8.12 PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear Register

Disables the generation of interrupt requests on overflows from the Cycle Count Register, PMU.PMCCNTR_ELO, and the event counters PMU.PMEVCNTR<n>_ELO. Reading the register shows which overflow interrupt requests are enabled.

Configurations

PMINTENCLR_EL1 is in the Core power domain.

External register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.2 PMINTENCLR_EL1, Performance Monitors Interrupt Enable Clear Register](#) on page 713 bits [31:0].

External register PMINTENCLR_EL1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.1 PMINTENSET_EL1, Performance Monitors Interrupt Enable Set Register](#) on page 710 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xC60

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-466: PMU_PMINTENCLR_EL1 bit assignments

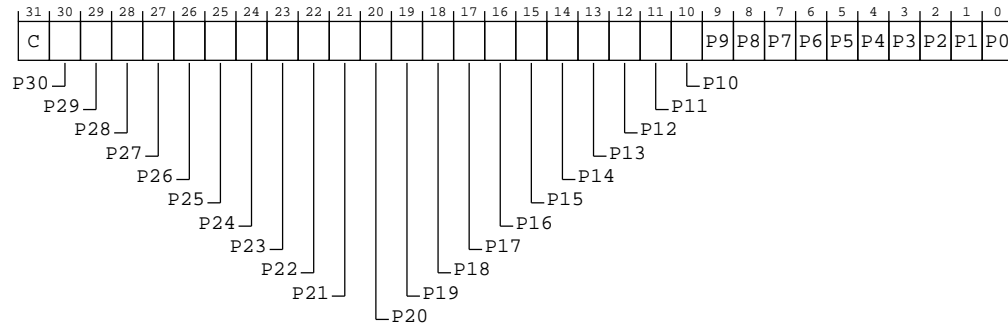


Table B-915: PMINTENCLR_EL1 bit descriptions

Bits	Name	Description	Reset
[31]	C	<p>Interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO disable. On writes, allows software to disable the interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO. On reads, returns the interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO enable status.</p> <p>0b0</p> <p>Interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO disabled.</p> <p>0b1</p> <p>Interrupt request or PMU exception on unsigned overflow of PMU.PMCCNTR_ELO enabled.</p> <p>Access to this field is: W1C</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO disable. On writes, allows software to disable the interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO. On reads, returns the interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO enable status.</p> <p>0b0</p> <p>Interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO disabled.</p> <p>0b1</p> <p>Interrupt request or PMU exception on unsigned overflow of PMEVCNTR<m>_ELO enabled.</p> <p>When m >= 6</p> <p>Access to this field is: RAZ/WI</p> <p>Otherwise</p> <p>Access to this field is: W1C</p>	31 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xC60	PMINTENCLR_EL1	31:0

B.2.2.8.13 PMOVSLR_ELO, Performance Monitors Overflow Flag Status Clear register

Contains the state of the overflow bit for the Cycle Count Register, PMU.PMCCNTR_ELO, and each of the implemented event counters AArch32-PMEVCNTR<n>. Writing to this register clears these bits.

Configurations

PMOVSLR_ELO is in the Core power domain.

External register PMOVSLR_ELO bits [31:0] are architecturally mapped to AArch64 System register A.2.3.22 PMOVSLR_ELO, Performance Monitors Overflow Flag Status Clear Register on page 768 bits [31:0].

External register PMOVSLR_ELO bits [31:0] are architecturally mapped to AArch64 System register A.2.3.31 PMOVSET_ELO, Performance Monitors Overflow Flag Status Set Register on page 803 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xC80

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-467: PMU_PMOVSLR_ELO bit assignments

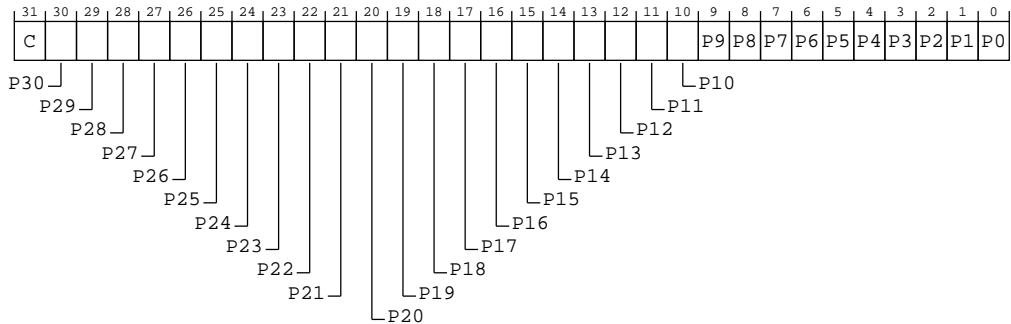


Table B-917: PMOVSLR_ELO bit descriptions

Bits	Name	Description	Reset
[31]	C	Unsigned overflow flag for PMU.PMCCNTR_ELO clear. On writes, allows software to clear the unsigned overflow flag for PMU.PMCCNTR_ELO to 0. On reads, returns the unsigned overflow flag for PMU.PMCCNTR_ELO overflow status. 0b0 PMU.PMCCNTR_ELO has not overflowed. 0b1 PMU.PMCCNTR_ELO has overflowed. Access to this field is: W1C	x
[30:0]	P<m>, bit[m], where m = 30 to 0	Unsigned overflow flag for PMU.PMEVCNTR<m>_ELO clear. On writes, allows software to clear the unsigned overflow flag for PMU.PMEVCNTR<m>_ELO to 0. On reads, returns the unsigned overflow flag for PMU.PMEVCNTR<m>_ELO overflow status. 0b0 PMU.PMEVCNTR<m>_ELO has not overflowed. 0b1 PMU.PMEVCNTR<m>_ELO has overflowed. When m >= 6 Access to this field is: RAZ/WI Otherwise Access to this field is: W1C	31 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xC80	PMOVSLR_ELO	31:0

B.2.2.8.14 PMSWINC_ELO, Performance Monitors Software Increment Register

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see *SW_INCR* in the [Arm® Architecture Reference Manual for A-profile architecture](#).

Configurations

Use of this register is deprecated.

External register PMSWINC_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.23 PMSWINC_ELO, Performance Monitors Software Increment Register](#) on page 771 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xCA0

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-468: PMU_PMSWINC_ELO bit assignments

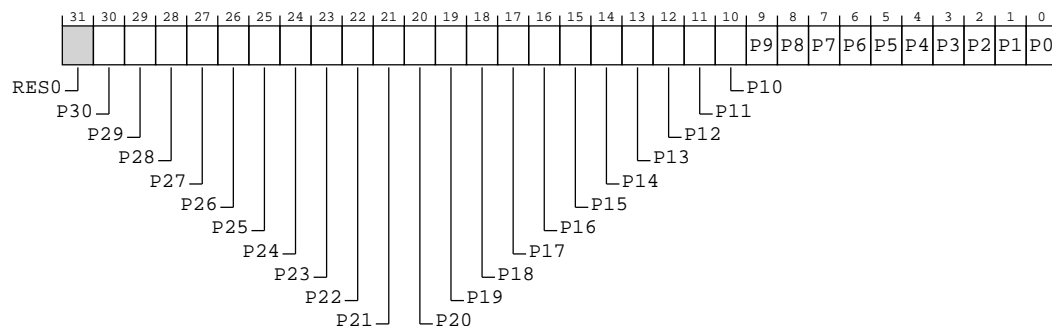


Table B-919: PMSWINC_ELO bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	RES0
[30:0]	P<n>, bit[n], where n = 30 to 0	<p>Event counter software increment bit for PMU.PMEVCNTR<n>_ELO.</p> <p>If PMU.PMCFGR.N is less than 31, bits [30:PMU.PMCFGR.N] are wI.</p> <p>0b0 No action. The write to this bit is ignored.</p> <p>0b1 A SW_INCR event is generated for event counter n.</p>	31 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xCA0	PMSWINC_ELO	None

B.2.2.8.15 PMOVSSET_ELO, Performance Monitors Overflow Flag Status Set Register

Sets the state of the overflow bit for the Cycle Count Register, PMU.PMCCNTR_ELO, and each of the implemented event counters PMU.PMEVCNTR<n>_ELO.

Configurations

PMOVSSET_ELO is in the Core power domain.

External register PMOVSSET_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.31 PMOVSSET_ELO, Performance Monitors Overflow Flag Status Set Register](#) on page 803 bits [31:0].

External register PMOVSSET_ELO bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.22 PMOVSCCLR_ELO, Performance Monitors Overflow Flag Status Clear Register](#) on page 768 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xCC0

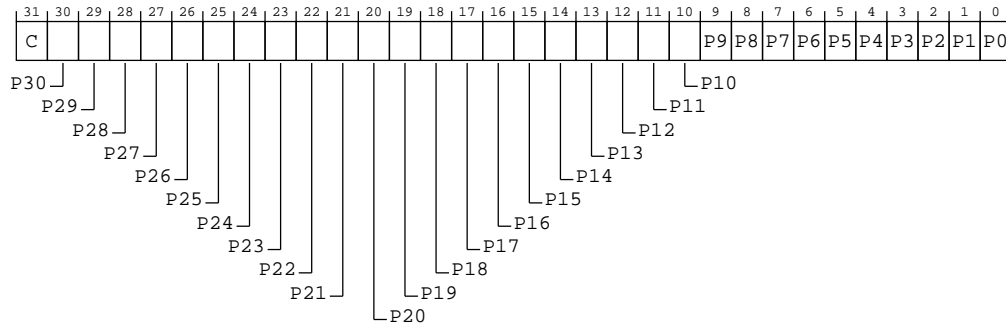
Reset value

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-469: PMU_PMOVSSET_ELO bit assignments**Table B-921: PMOVSSET_ELO bit descriptions**

Bits	Name	Description	Reset
[31]	C	<p>Unsigned overflow flag for PMU.PMCCNTR_ELO set. On writes, allows software to set the unsigned overflow flag for PMU.PMCCNTR_ELO to 1. On reads, returns the unsigned overflow flag for PMU.PMCCNTR_ELO overflow status.</p> <p>0b0</p> <p>PMU.PMCCNTR_ELO has not overflowed.</p> <p>0b1</p> <p>PMU.PMCCNTR_ELO has overflowed.</p> <p>Access to this field is: W1S</p>	x
[30:0]	P<m>, bit[m], where m = 30 to 0	<p>Unsigned overflow flag for PMU.PMEVCNTR<m>_ELO set. On writes, allows software to set the unsigned overflow flag for PMU.PMEVCNTR<m>_ELO to 1. On reads, returns the unsigned overflow flag for PMU.PMEVCNTR<m>_ELO overflow status.</p> <p>0b0</p> <p>PMU.PMEVCNTR<m>_ELO has not overflowed.</p> <p>0b1</p> <p>PMU.PMEVCNTR<m>_ELO has overflowed.</p> <p>When m >= 6</p> <p>Access to this field is: RAZ/WI</p> <p>Otherwise</p> <p>Access to this field is: W1S</p>	31 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xCC0	PMOVSSET_ELO	31:0

B.2.2.8.16 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

Reset value

0000	xxxx	xxxx	0001	0111	1111	0000	0110
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-470: PMU_PMCFG bit assignments

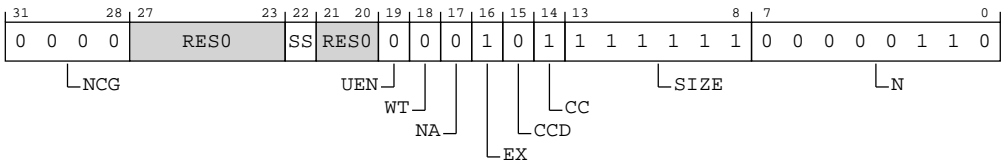


Table B-923: PMCFGR bit descriptions

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. If the architecture-defined form of snapshot is not implemented, a PMU might include an IMPLEMENTATION DEFINED snapshot mechanism, including one using the IMPLEMENTATION DEFINED registers 0x600-0x7FC and 0xE30-0xE3C.	The reset values can be the following: 0b0, 0b1, respective to the value.
[21:20]	RES0	Reserved	RES0
[19]	UEN	User-mode Enable Register supported. AArch64-PMUSERENR_ELO is not visible in the external debug interface, so this bit is RAZ . 0b0	0b0
[18]	WT	This feature is not supported, so this bit is RAZ . 0b0	0b0
[17]	NA	This feature is not supported, so this bit is RAZ . 0b0	0b0
[16]	EX	Export supported. Value is IMPLEMENTATION DEFINED . 0b1 PMU.PMCR_ELO.X is read/write.	0b1
[15]	CCD	Cycle counter has prescale. This field is RAZ 0b0 PMU.PMCR_ELO.D is RES0 .	0b0
[14]	CC	Dedicated cycle counter (counter 31) supported. 0b1	0b1

Bits	Name	Description	Reset
[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_ELO. 0b00000110 ext-PMCCNTR_ELO plus 6 event counters implemented.	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

B.2.2.8.17 PMCR_ELO, 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_ELO is in the Core power domain.

External register PMCR_ELO bits [63:32, 10:0] are architecturally mapped to AArch64 System register [A.2.3.19 PMCR_ELO, Performance Monitors Control Register](#) on page 759 bits [63:32].

Attributes

Width

32

Component

PMU

Register offset

0xE04

Reset value

```
0000 0000 0000 0000 0000 0xxx xxx0 x000
```




Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-471: PMU_PMCR_ELO bit assignments



Table B-925: PMCR_ELO bit descriptions

Bits	Name	Description	Reset
[31:11]	RAZ/WI	Reserved	RAZ/WI
[10:7]	RES0	Reserved	RES0
[6]	RES1	Reserved	RES1
[5]	DP	<p>Disable cycle counter when event counting is prohibited. The possible values of this bit are:</p> <p>0b0</p> <p>Cycle counting by PMU.PMCCNTR_ELO is not affected by this mechanism.</p> <p>0b1</p> <p>Cycle counting by PMU.PMCCNTR_ELO is disabled in prohibited regions and when event counting is frozen:</p> <ul style="list-style-type: none"> If FEAT_PMUv3p1 is implemented, EL2 is implemented, and AArch64-MDCR_EL2.HPMD is 1, then cycle counting by PMU.PMCCNTR_ELO is disabled at EL2. <p>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.</p> <p>For more information, see <i>Prohibiting event and cycle counting</i> in the Arm® Architecture Reference Manual for A-profile architecture.</p>	x
[4]	X	<p>Enable export of events to the core trace unit (ETM).</p> <p>0b0</p> <p>Do not export events.</p> <p>0b1</p> <p>Export events to the ETM, where not prohibited.</p>	0b0
[3]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[2]	C	<p>Cycle counter reset. The effects of writing to this bit are:</p> <p>0b0</p> <p>No action.</p> <p>0b1</p> <p>Reset PMU.PMCCNTR_ELO to zero.</p> <p>Resetting PMU.PMCCNTR_ELO does not change the cycle counter overflow bit.</p> <p>Access to this field is: WO/RAZ</p>	0b0
[1]	P	<p>Event counter reset. The effects of writing to this bit are:</p> <p>0b0</p> <p>No action.</p> <p>0b1</p> <p>Reset all event counters, not including PMU.PMCCNTR_ELO, to zero.</p> <p>Resetting the event counters does not change the event counter overflow bits.</p> <p>Access to this field is: WO/RAZ</p>	0b0
[0]	E	<p>Enable.</p> <p>In the description of this field:</p> <ul style="list-style-type: none"> If EL2 is implemented and is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. <p>0b0</p> <p>PMU.PMCCNTR_ELO is disabled and event counters PMU.PMEVCNTR<n>_ELO, where n is in the range of affected event counters, are disabled.</p> <p>0b1</p> <p>PMU.PMCCNTR_ELO and event counters PMU.PMEVCNTR<n>_ELO, where n is in the range of affected event counters, are enabled by PMU.PMCNTENSET_ELO.</p> <p>If PMN is not 0, this field affects the operation of event counters in the range [0 .. (PMN-1)].</p> <p>This field does not affect the operation of other event counters.</p>	0b0

Accessibility

Component	Offset	Instance	Range
PMU	0xE04	PMCR_ELO	None

B.2.2.8.18 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](#).



This view of the register was previously called PMCEID0_ELO.

Configurations

PMCEID0 is in the Core power domain.

External register PMCEID0 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.25 PMCEID0_ELO, Performance Monitors Common Event Identification Register 0](#) on page 775 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xE20

Reset value

1111 1110 0011 1111 1111 1111 1111 1111

Bit descriptions

Figure B-472: PMU_PMCEID0 bit assignments

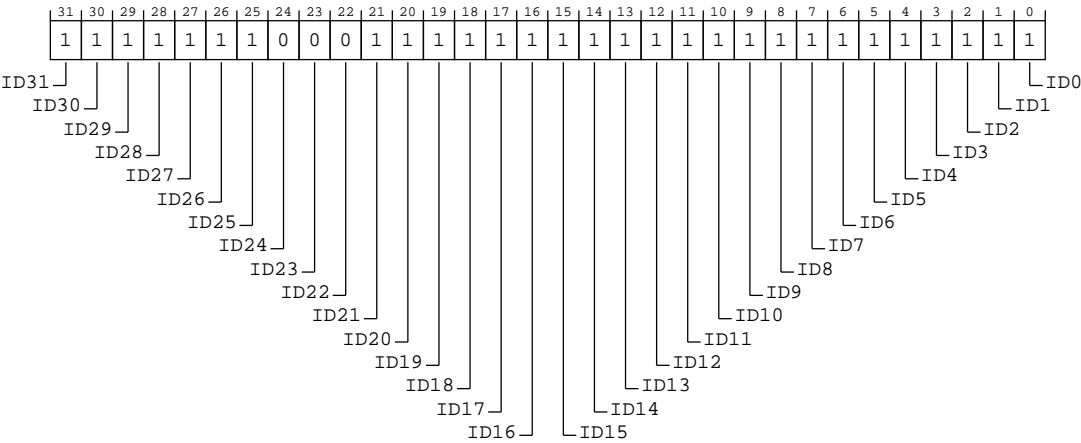


Table B-927: PMCEID0 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE_ALLOCATE event implemented.	0b1
[30]	ID30	ID[n] corresponds to Common event n. For each bit: 0b1 CHAIN event implemented.	0b1
[29]	ID29	ID[n] corresponds to Common event n. For each bit: 0b1 BUS_CYCLES event implemented.	0b1
[28]	ID28	ID[n] corresponds to Common event n. For each bit: 0b1 TTBR_WRITE_RETIRED event implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event n. For each bit: 0b1 INST_SPEC event implemented.	0b1
[26]	ID26	ID[n] corresponds to Common event n. For each bit: 0b1 MEMORY_ERROR event implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event n. For each bit: 0b1 BUS_ACCESS event implemented.	0b1
[24]	ID24	ID[n] corresponds to Common event n. For each bit: 0b0 Event 0x0018 not implemented.	0b0
[23]	ID23	ID[n] corresponds to Common event n. For each bit: 0b0 Event 0x0017 not implemented.	0b0

Bits	Name	Description	Reset
[22]	ID22	ID[n] corresponds to Common event n. For each bit: 0b0 Event 0x0016 not implemented.	0b0
[21]	ID21	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE_WB event implemented.	0b1
[20]	ID20	ID[n] corresponds to Common event n. For each bit: 0b1 L1I_CACHE event implemented.	0b1
[19]	ID19	ID[n] corresponds to Common event n. For each bit: 0b1 MEM_ACCESS event implemented.	0b1
[18]	ID18	ID[n] corresponds to Common event n. For each bit: 0b1 BR_PRED event implemented.	0b1
[17]	ID17	ID[n] corresponds to Common event n. For each bit: 0b1 CPU_CYCLES event implemented.	0b1
[16]	ID16	ID[n] corresponds to Common event n. For each bit: 0b1 BR_MIS_PRED event implemented.	0b1
[15]	ID15	ID[n] corresponds to Common event n. For each bit: 0b1 UNALIGNED_LDST_RETIRED event implemented.	0b1
[14]	ID14	ID[n] corresponds to Common event n. For each bit: 0b1 BR_RETURN_RETIRED event implemented.	0b1

Bits	Name	Description	Reset
[13]	ID13	ID[n] corresponds to Common event n. For each bit: 0b1 BR_IMMED_RETIRED event implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event n. For each bit: 0b1 PC_WRITE_RETIRED event implemented.	0b1
[11]	ID11	ID[n] corresponds to Common event n. For each bit: 0b1 CID_WRITE_RETIRED event implemented.	0b1
[10]	ID10	ID[n] corresponds to Common event n. For each bit: 0b1 EXC_RETURN event implemented.	0b1
[9]	ID9	ID[n] corresponds to Common event n. For each bit: 0b1 EXC_TAKEN event implemented.	0b1
[8]	ID8	ID[n] corresponds to Common event n. For each bit: 0b1 INST_RETIRED event implemented.	0b1
[7]	ID7	ID[n] corresponds to Common event n. For each bit: 0b1 ST_RETIRED event implemented.	0b1
[6]	ID6	ID[n] corresponds to Common event n. For each bit: 0b1 LD_RETIRED event implemented.	0b1
[5]	ID5	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_TLB_REFILL event implemented.	0b1

Bits	Name	Description	Reset
[4]	ID4	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE event implemented.	0b1
[3]	ID3	ID[n] corresponds to Common event n. For each bit: 0b1 L1D_CACHE_REFILL event implemented.	0b1
[2]	ID2	ID[n] corresponds to Common event n. For each bit: 0b1 L1I_TLB_REFILL event implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event n. For each bit: 0b1 L1I_CACHE_REFILL event implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event n. For each bit: 0b1 SW_INCR event 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

B.2.2.8.19 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](#).



This view of the register was previously called PMCEID1_ELO.

Configurations

PMCEID1 is in the Core power domain.

External register PMCEID1 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.26 PMCEID1_ELO](#), [Performance Monitors Common Event Identification Register 1](#) on page 784 bits [31:0].

Attributes

Width

32

Component

PMU

Register offset

0xE24

Reset value

1111 1111 1100 0000 1010 0000 0111 1111

Bit descriptions

Figure B-473: PMU_PMCEID1 bit assignments

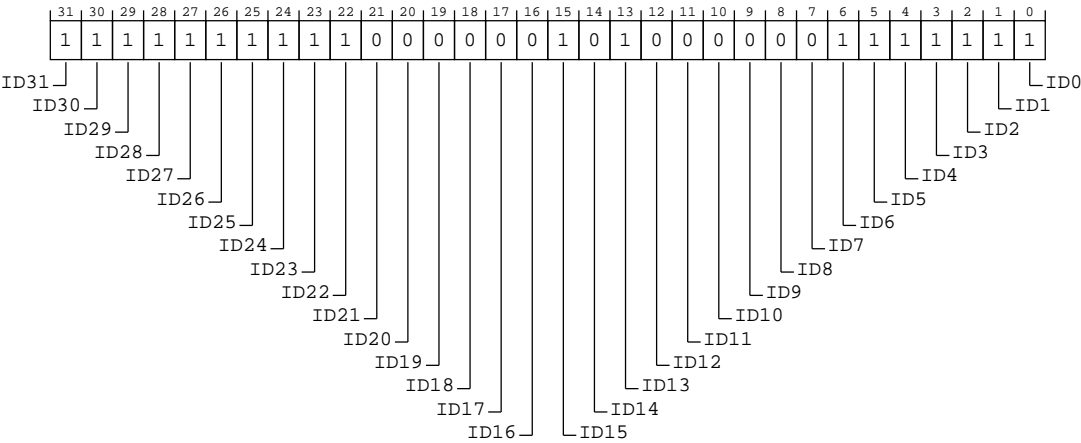


Table B-929: PMCEID1 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_SLOT event implemented.	0b1
[30]	ID30	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_SLOT_FRONTEND event implemented.	0b1
[29]	ID29	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_SLOT_BACKEND event implemented.	0b1
[28]	ID28	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL event implemented.	0b1
[27]	ID27	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 OP_SPEC event implemented.	0b1
[26]	ID26	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 OP_RETIRED event implemented.	0b1
[25]	ID25	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L1D_CACHE_LMISS_RD event implemented.	0b1
[24]	ID24	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 REMOTE_ACCESS_RD event implemented.	0b1
[23]	ID23	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 LL_CACHE_MISS_RD event implemented.	0b1

Bits	Name	Description	Reset
[22]	ID22	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 LL_CACHE_RD event implemented.	0b1
[21]	ID21	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0035 not implemented.	0b0
[20]	ID20	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0034 not implemented.	0b0
[19]	ID19	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0033 not implemented.	0b0
[18]	ID18	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0032 not implemented.	0b0
[17]	ID17	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0031 not implemented.	0b0
[16]	ID16	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0030 not implemented.	0b0
[15]	ID15	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L2D_TLB event implemented.	0b1
[14]	ID14	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002E not implemented.	0b0

Bits	Name	Description	Reset
[13]	ID13	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L2D_TLB_REFILL event implemented.	0b1
[12]	ID12	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002C not implemented.	0b0
[11]	ID11	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002B not implemented.	0b0
[10]	ID10	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x002A not implemented.	0b0
[9]	ID9	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0029 not implemented.	0b0
[8]	ID8	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0028 not implemented.	0b0
[7]	ID7	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b0 Event 0x0027 not implemented.	0b0
[6]	ID6	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L1I_TLB event implemented.	0b1
[5]	ID5	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L1D_TLB event implemented.	0b1

Bits	Name	Description	Reset
[4]	ID4	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_BACKEND event implemented.	0b1
[3]	ID3	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 STALL_FRONTEND event implemented.	0b1
[2]	ID2	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 BR_MIS_PRED_RETIRED event implemented.	0b1
[1]	ID1	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 BR_RETIRED event implemented.	0b1
[0]	ID0	ID[n] corresponds to Common event (0x0020 + n). For each bit: 0b1 L2D_CACHE_ALLOCATE event 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

B.2.2.8.20 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.

External register PMCEID2 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.25 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0](#) on page 775 bits [63:32].

Attributes

Width

32

Component

PMU

Register offset

0xE28

Reset value

0000 0000 0000 0000 0000 0000 0100 0000

Bit descriptions

Figure B-474: PMU_PMCEID2 bit assignments

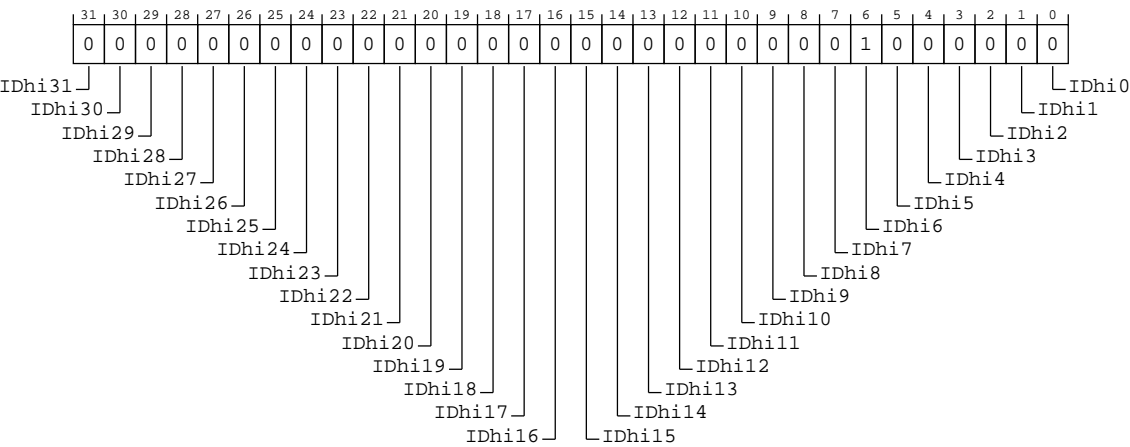


Table B-931: PMCEID2 bit descriptions

Bits	Name	Description	Reset
[31]	IDHi31	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401F not implemented.	0b0

Bits	Name	Description	Reset
[30]	IDhi30	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401E not implemented.	0b0
[29]	IDhi29	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401D not implemented.	0b0
[28]	IDhi28	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401C not implemented.	0b0
[27]	IDhi27	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401B not implemented.	0b0
[26]	IDhi26	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x401A not implemented.	0b0
[25]	IDhi25	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4019 not implemented.	0b0
[24]	IDhi24	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4018 not implemented.	0b0
[23]	IDhi23	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4017 not implemented.	0b0
[22]	IDhi22	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4016 not implemented.	0b0

Bits	Name	Description	Reset
[21]	IDhi21	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4015 not implemented.	0b0
[20]	IDhi20	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4014 not implemented.	0b0
[19]	IDhi19	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4013 not implemented.	0b0
[18]	IDhi18	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4012 not implemented.	0b0
[17]	IDhi17	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4011 not implemented.	0b0
[16]	IDhi16	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4010 not implemented.	0b0
[15]	IDhi15	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400F not implemented.	0b0
[14]	IDhi14	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400E not implemented.	0b0
[13]	IDhi13	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400D not implemented.	0b0

Bits	Name	Description	Reset
[12]	IDhi12	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400C not implemented.	0b0
[11]	IDhi11	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400B not implemented.	0b0
[10]	IDhi10	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x400A not implemented.	0b0
[9]	IDhi9	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4009 not implemented.	0b0
[8]	IDhi8	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4008 not implemented.	0b0
[7]	IDhi7	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4007 not implemented.	0b0
[6]	IDhi6	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b1 L1I_CACHE_LMISS event implemented.	0b1
[5]	IDhi5	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4005 not implemented.	0b0
[4]	IDhi4	IDhi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4004 not implemented.	0b0

Bits	Name	Description	Reset
[3]	IDHi3	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4003 not implemented.	0b0
[2]	IDHi2	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4002 not implemented.	0b0
[1]	IDHi1	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4001 not implemented.	0b0
[0]	IDHi0	IDHi[n] corresponds to Common event (0x4000 + n). For each bit: 0b0 Event 0x4000 not implemented.	0b0

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

B.2.2.8.21 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.

External register PMCEID3 bits [31:0] are architecturally mapped to AArch64 System register [A.2.3.26 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1](#) on page 784 bits [63:32].

Attributes

Width

32

Component

PMU

Register offset

0xE2C

Reset value

0000 0000 0000 0000 0000 0000 0000 0111

Bit descriptions

Figure B-475: PMU_PMCEID3 bit assignments

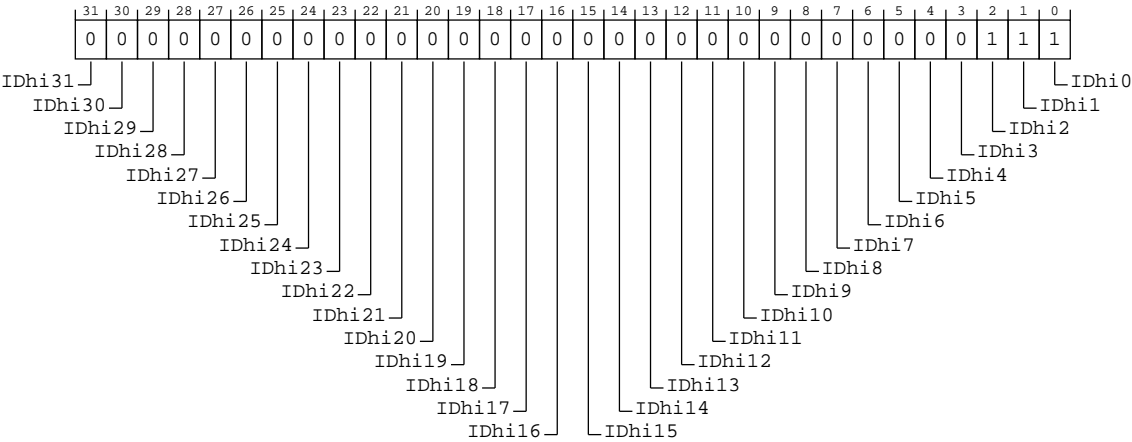


Table B-933: PMCEID3 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403F not implemented.	0b0
[30]	IDhi30	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403E not implemented.	0b0

Bits	Name	Description	Reset
[29]	IDHi29	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403D not implemented.	0b0
[28]	IDHi28	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403C not implemented.	0b0
[27]	IDHi27	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403B not implemented.	0b0
[26]	IDHi26	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x403A not implemented.	0b0
[25]	IDHi25	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4039 not implemented.	0b0
[24]	IDHi24	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4038 not implemented.	0b0
[23]	IDHi23	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4037 not implemented.	0b0
[22]	IDHi22	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4036 not implemented.	0b0
[21]	IDHi21	IDHi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4035 not implemented.	0b0

Bits	Name	Description	Reset
[20]	IDhi20	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4034 not implemented.	0b0
[19]	IDhi19	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4033 not implemented.	0b0
[18]	IDhi18	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4032 not implemented.	0b0
[17]	IDhi17	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4031 not implemented.	0b0
[16]	IDhi16	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4030 not implemented.	0b0
[15]	IDhi15	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402F not implemented.	0b0
[14]	IDhi14	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402E not implemented.	0b0
[13]	IDhi13	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402D not implemented.	0b0
[12]	IDhi12	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402C not implemented.	0b0

Bits	Name	Description	Reset
[11]	IDhi11	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402B not implemented.	0b0
[10]	IDhi10	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x402A not implemented.	0b0
[9]	IDhi9	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4029 not implemented.	0b0
[8]	IDhi8	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4028 not implemented.	0b0
[7]	IDhi7	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4027 not implemented.	0b0
[6]	IDhi6	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4026 not implemented.	0b0
[5]	IDhi5	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4025 not implemented.	0b0
[4]	IDhi4	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4024 not implemented.	0b0
[3]	IDhi3	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b0 Event 0x4023 not implemented.	0b0

Bits	Name	Description	Reset
[2]	IDhi2	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b1 ST_ALIGN_LAT event implemented.	0b1
[1]	IDhi1	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b1 LD_ALIGN_LAT event implemented.	0b1
[0]	IDhi0	IDhi[n] corresponds to Common event (0x4020 + n). For each bit: 0b1 LDST_ALIGN_LAT event 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

B.2.2.8.22 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

Reset value

xxxx	xxxx	xxxx	0110	0000	0010	0000	0011
31	27	23	19	15	11	7	3



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-476: PMU_PMMIR bit assignments

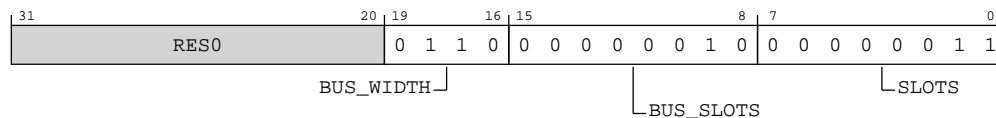


Table B-935: PMMIR bit descriptions

Bits	Name	Description	Reset
[31:20]	RES0	Reserved	RES0
[19:16]	BUS_WIDTH	<p>Bus width. Indicates the number of bytes each BUS_ACCESS event relates to. Encoded as $\log_2(\text{number of bytes})$, plus one.</p> <p>0b0110 32 bytes.</p> <p>All other values are reserved.</p> <p>Each transfer is up to this number of bytes. An access might be smaller than the bus width.</p> <p>When this field is nonzero, each access counted by BUS_ACCESS is at most BUS_WIDTH bytes. An implementation might treat a wide bus as multiple narrower buses, such that a wide access on the bus increments the BUS_ACCESS counter by more than one.</p>	0b0110
[15:8]	BUS_SLOTS	<p>Bus count. The largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle.</p> <p>0b00000010 The BUS_ACCESS event might increment by at most 2 in a single BUS_CYCLES cycle.</p> <p>When this field is nonzero, the largest value by which the BUS_ACCESS event might increment in a single BUS_CYCLES cycle is BUS_SLOTS.</p> <p>If the bus count information is not available, this field will read as zero.</p>	0x02
[7:0]	SLOTS	<p>Operation width. The largest value by which the STALL_SLOT event might increment in a single cycle.</p> <p>0b00000011 STALL_SLOT may increment by up to 3 in a single cycle.</p>	0x03

Accessibility

If the Core power domain is off or in a low-power state, access on this interface returns an Error.

Component	Offset	Instance	Range
PMU	0xE40	PMMIR	31:0

B.2.2.8.23 PMITCTRL, Performance Monitors Integration mode Control register

Enables the Performance Monitors to switch from default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

Configurations

This register is in the Core power domain.

Attributes

Width

32

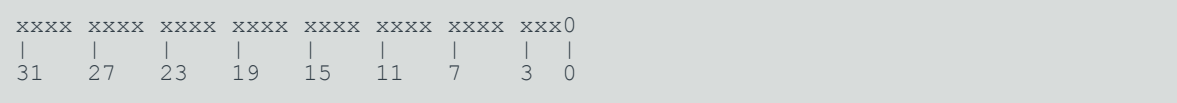
Component

PMU

Register offset

0xF00

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-477: PMU_PMITCTRL bit assignments

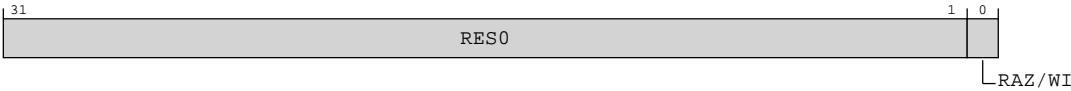


Table B-937: PMITCTRL bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[0]	RAZ/WI	Reserved	RAZ/WI

Accessibility

Component	Offset	Instance	Range
PMU	0xF00	PMITCTRL	None

B.2.2.8.24 PMDEVAFF0, Performance Monitors Device Affinity register 0

Copy of the low half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFA8

Reset value



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-478: PMU_PMDEVAFF0 bit assignments

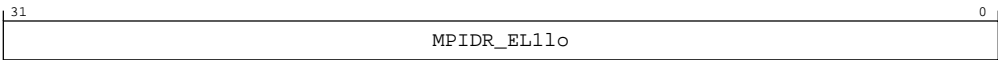


Table B-939: PMDEVAFF0 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1lo	AArch64-MPIDR_EL1 low half. Read-only copy of the low half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xFA8	PMDEVAFF0	None

B.2.2.8.25 PMDEVAFF1, Performance Monitors Device Affinity register 1

Copy of the high half of the PE AArch64-MPIDR_EL1 register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFAC

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-479: PMU_PMDEVAFF1 bit assignments

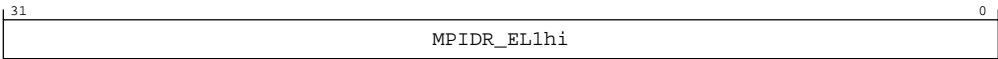


Table B-941: PMDEVAFF1 bit descriptions

Bits	Name	Description	Reset
[31:0]	MPIDR_EL1hi	AArch64-MPIDR_EL1 high half. Read-only copy of the high half of AArch64-MPIDR_EL1, as seen from the highest implemented Exception level.	32 {x}

Accessibility

Component	Offset	Instance	Range
PMU	0xFAC	PMDEVAFF1	None

B.2.2.8.26 PMLAR, Performance Monitors Lock Access Register

Allows or disallows access to the Performance Monitors registers through a memory-mapped interface.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFB0

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

Otherwise

Figure B-480: PMU_PMLAR bit assignments

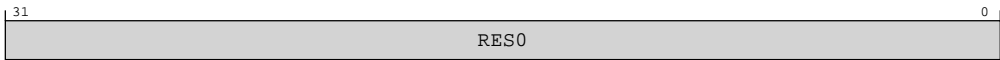


Table B-943: PMLAR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	RES0

Accessibility

Component	Offset	Instance	Range
PMU	0xFB0	PMLAR	None

B.2.2.8.27 PMLSR, Performance Monitors Lock Status Register

Indicates the current status of the software lock for Performance Monitors registers.

The optional Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Configurations

Software Lock is not implemented by the architecturally-defined debug components of the PE in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFB4

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	x000
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-481: PMU_PMLSR bit assignments

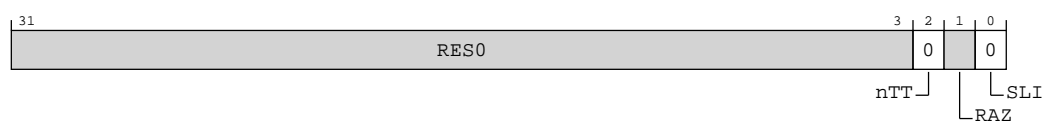


Table B-945: PMLSR bit descriptions

Bits	Name	Description	Reset
[31:3]	RES0	Reserved	RES0
[2]	nTT	Not thirty-two bit access required. 0b0	0b0
[1]	RAZ	Reserved	RAZ
[0]	SLI	Indicates whether the Software Lock is implemented. 0b0 Software Lock not implemented or not memory-mapped access.	0b0

Accessibility

Component	Offset	Instance	Range
PMU	0xFB4	PMLSR	None

B.2.2.8.28 PMAUTHSTATUS, Performance Monitors Authentication Status register

Provides information about the state of the **IMPLEMENTATION DEFINED** authentication interface for Performance Monitors.

Configurations

This register is in the Core power domain.

Attributes

Width

32

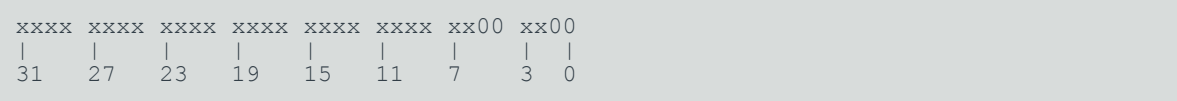
Component

PMU

Register offset

0xFB8

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-482: PMU_PMAUTHSTATUS bit assignments

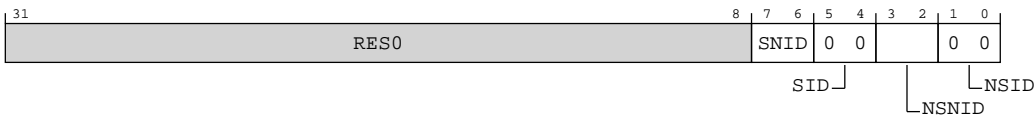


Table B-947: PMAUTHSTATUS bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:6]	SNID	Holds the same value as ext-DBGAUTHSTATUS_EL1.SNID.	xx
[5:4]	SID	Secure invasive debug. Possible values of this field are: 0b00 Not implemented.	0b00
[3:2]	NSNID	Holds the same value as ext-DBGAUTHSTATUS_EL1.NSNID.	xx
[1:0]	NSID	Non-secure invasive debug. Possible values of this field are: 0b00 Not implemented.	0b00

Accessibility

Component	Offset	Instance	Range
PMU	0xFB8	PMAUTHSTATUS	None

B.2.2.8.29 PMDEVARCH, Performance Monitors Device Architecture register

Identifies the programmers' model architecture of the Performance Monitor component.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFBC

Reset value

0100 0111 0111 0000 0010 1010 0001 0110

Bit descriptions

Figure B-483: PMU_PMDEVARCH bit assignments

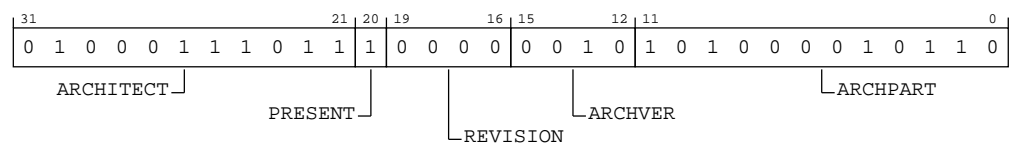


Table B-949: PMDEVARCH bit descriptions

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. PMDEVARCH.ARCHVER and PMDEVARCH.ARCHPART are also defined as a single field, PMDEVARCH.ARCHID, so that PMDEVARCH.ARCHPART is PMDEVARCH.ARCHID[11:0].	0xA16

Accessibility

Component	Offset	Instance	Range
PMU	0xFBC	PMDEVARCH	None

B.2.2.8.30 PMDEVID, Performance Monitors Device ID register

Provides information about features of the Performance Monitors implementation.

Configurations

This register is in the Core power domain.

Attributes

Width

32

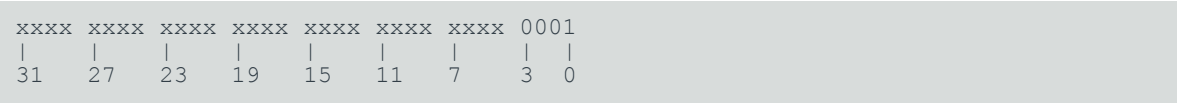
Component

PMU

Register offset

0xFC8

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-484: PMU_PMDEVID bit assignments



Table B-951: PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	RES0
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. 0b0001 PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	0b0001

Accessibility

Component	Offset	Instance	Range
PMU	0xFC8	PMDEVID	None

B.2.2.8.31 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFCC

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	0110
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-485: PMU_PMDEVTYPE bit assignments

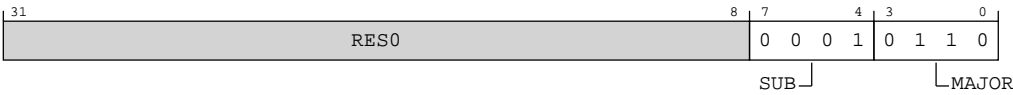


Table B-953: PMDEVTYPE bit descriptions

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

B.2.2.8.32 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

This register is in the Core power domain.

Attributes

Width

32

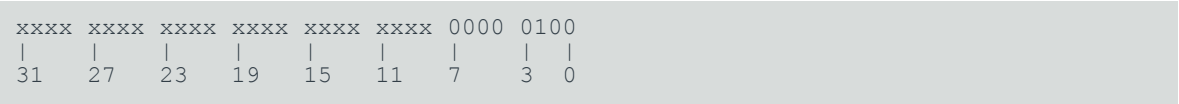
Component

PMU

Register offset

0xFD0

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-486: PMU_PMPIDR4 bit assignments

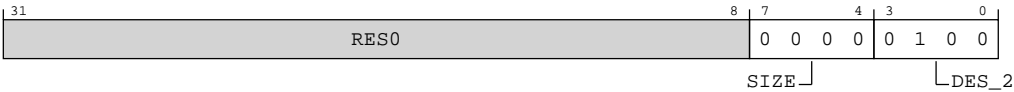


Table B-955: PMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	SIZE	Size of the component. RAZ. Log ₂ of the number of 4KB pages from the start of the component to the end of the component ID registers. 0b0000 The component uses a single 4KB block.	0b0000
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. 0b0100 Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B.	0b0100

Accessibility

Component	Offset	Instance	Range
PMU	0xFD0	PMPIDR4	None

B.2.2.8.33 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

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFE0

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	0001	0101
31	27	23	19	15	11	7	3 0



Note

Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-487: PMU_PMPIDR0 bit assignments

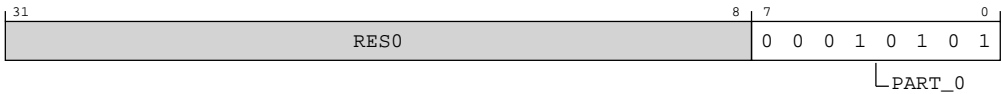


Table B-957: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PART_0	Part number bits [7:0]. 0b00010101 Cortex-R82 PMU. Bits [7:0] of part number 0xD15.	0x15

Accessibility

Component	Offset	Instance	Range
PMU	0xFE0	PMPIDR0	None

B.2.2.8.34 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

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFE4

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-488: PMU_PMPIDR1 bit assignments

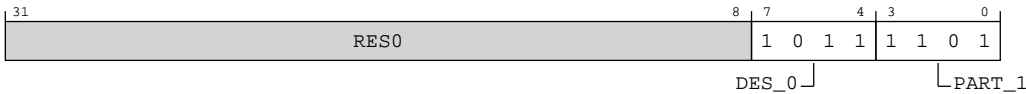


Table B-959: PMPIDR1 bit descriptions

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. Bits [3:0] of JEP106 identification code 0x3B.	0b1011
[3:0]	PART_1	Part number bits [11:8]. 0b1101 Cortex-R82 PMU. Bits [11:8] of part number 0xD15.	0b1101

Accessibility

Component	Offset	Instance	Range
PMU	0xFE4	PMPIDR1	None

B.2.2.8.35 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

This register is in the Core power domain.

Attributes

Width

32

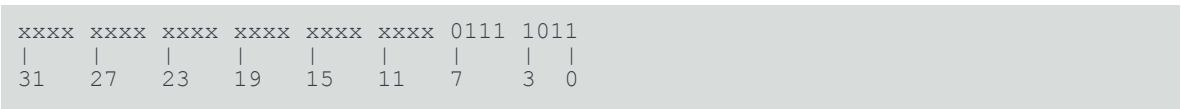
Component

PMU

Register offset

0xFE8

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-489: PMU_PMPIDR2 bit assignments

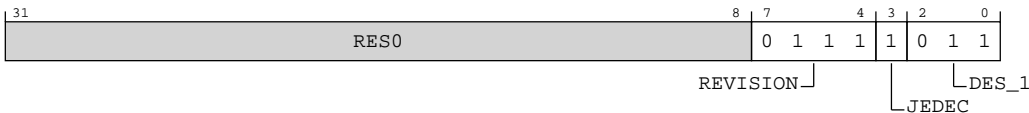


Table B-961: PMPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVISION	Component revision. 0b0111 Revision 7.	0b0111
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. 0b1 JEDEC-assignee values is used.	0b1
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. 0b011 Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	0b011

Accessibility

Component	Offset	Instance	Range
PMU	0xFE8	PMPIDR2	None

B.2.2.8.36 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

This register is in the Core power domain.

Attributes

Width

32

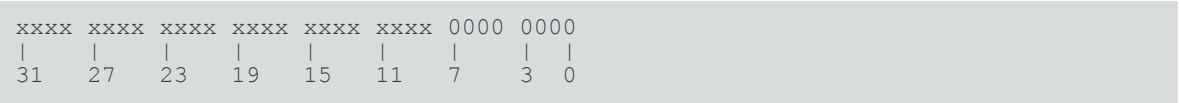
Component

PMU

Register offset

0xFEC

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-490: PMU_PMPIDR3 bit assignments



Table B-963: PMPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:4]	REVAND	Part minor revision. Parts using ext-PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. 0b0000 No ECO fixes.	0b0000
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. 0b0000 The component is not modified from the original design.	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFEC	PMPIDR3	None

B.2.2.8.37 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

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFF0

Reset value



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-491: PMU_PMCIDR0 bit assignments

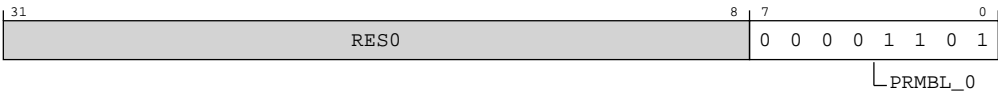


Table B-965: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_0	Preamble. 0b00001101 CoreSight component identification preamble.	0x0D

Accessibility

Component	Offset	Instance	Range
PMU	0xFF0	PMCIDR0	None

B.2.2.8.38 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

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFF4

Reset value

xxxx	xxxx	xxxx	xxxx	xxxx	xxxx	1001	0000
31	27	23	19	15	11	7	3 0



Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-492: PMU_PMCIDR1 bit assignments

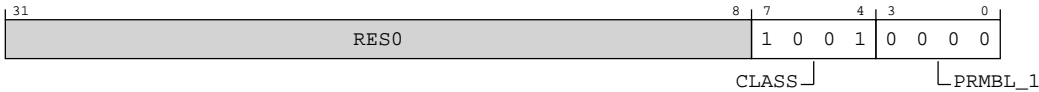


Table B-967: PMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0

Bits	Name	Description	Reset
[7:4]	CLASS	Component class. 0b1001 CoreSight debug component.	0b1001
[3:0]	PRMBL_1	Preamble. 0b0000 CoreSight component identification preamble.	0b0000

Accessibility

Component	Offset	Instance	Range
PMU	0xFF4	PMCIDR1	None

B.2.2.8.39 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

This register is in the Core power domain.

Attributes

Width

32

Component

PMU

Register offset

0xFF8

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-493: PMU_PMCIDR2 bit assignments



Table B-969: PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_2	Preamble. 0b00000101 CoreSight component identification preamble.	0x05

Accessibility

Component	Offset	Instance	Range
PMU	0xFF8	PMCIDR2	None

B.2.2.8.40 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

This register is in the Core power domain.

Attributes

Width

32

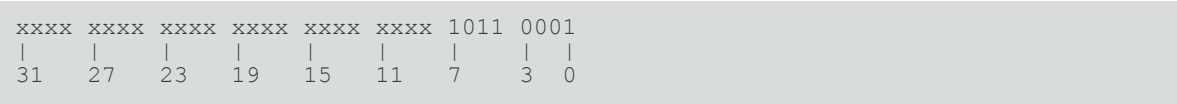
Component

PMU

Register offset

0xFFC

Reset value





Where the reset reads xxxx, see individual bits.

Bit descriptions

Figure B-494: PMU_PMCIDR3 bit assignments



Table B-971: PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	RES0
[7:0]	PRMBL_3	Preamble. 0b10110001 CoreSight component identification preamble.	0xB1

Accessibility

Component	Offset	Instance	Range
PMU	0xFFC	PMCIDR3	None

Appendix C Processor UNPREDICTABLE behaviors

This appendix describes the specific Cortex®-R82 processor **UNPREDICTABLE** behaviors that differ from the Arm standard behavior.

For each case, the Arm standard specification is listed under *Specification* and the Cortex®-R82 processor implementation is listed under *Implementation*. For detailed background information on **UNPREDICTABLE** behaviors, see [Arm® Architecture Reference Manual for A-profile architecture](#).

C.1 SBZ or SBO fields in instructions

This section describes the specification and implementation of SBZ or SBO fields in instructions.

Specification

Some instructions have (0) or (1) in the instruction decode to indicate *should-be-zero*, SBZ, or *should-be-one*, SBO. Except for specific cases identified in **CONSTRAINED UNPREDICTABLE** behaviors with Load-Exclusive/Store-Exclusive pairs as described in the [Arm® Architecture Reference Manual for A-profile architecture](#), if the instruction bit pattern of an instruction is executed with these fields not having the *should be* values, one of the following must occur:

- The instruction is **UNDEFINED**
- The instruction executes as a NOP
- The instruction operates as if the bit had the *should-be* value
- Any destination registers of the instruction become **UNKNOWN**
- For execution at EL0 or EL1, when EL2 is implemented and enabled for the current Security state and HCR_EL2.TIDCP is 1, the instruction is trapped to EL2 with EC value 0

Implementation

SBZ or SBO fields in instructions are treated as *don't care* conditions in the Cortex®-R82 decoders. Therefore, the instructions execute as if the bit is 1 or 0 if it is an SBO or SBZ value irrespective of the actual bit value.

C.2 CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values

This section describes the specification and implementation of caching of control or data values.

Specification

The Arm architecture allows copies of control values or data values to be cached in a cache or TLB. This can lead to **UNPREDICTABLE** behavior if the cache or TLB has not been correctly invalidated following a change of the control or data values.

Unless explicitly stated otherwise, the behavior of the PE is consistent with one of:

- The old data or control value.
- The new data or control value.
- An amalgamation of the old and new data or control values.

The **CONSTRAINED UNPREDICTABLE** case can arise from misprogramming when setting TTBR.CnP to 1, as identified in the descriptions of the TTBR.CnP field. In this case, for a particular TTBR, the behavior of the PE is consistent with one of:

- The value of the translation table entry pointed to by that TTBR on one of the PEs within the Inner Shareable domain for which both the value of TTBR.CnP is 1 and the other conditions for sharing translation table entries pointed to by that TTBR are met.
- An amalgamation of the values of the translation table entries pointed to by that TTBR on two or more of the PEs within the Inner Shareable domain for which both the value of TTBR.CnP is 1 and the other conditions for sharing translation table entries pointed to by that TTBR are met.

Implementation

For Cortex®-R82 processor, copies of control values are not stored in a cache. For cached copies of data values, old data value will be returned unless the cache is invalidated either explicitly or by the coherency mechanism. Copies of control values can be cached in TLB. Therefore, if invalidation is not performed after updating the control values, an erroneous translation may occur.

Cortex®-R82 processor does not store TTBR.CnP bit in the TLB as the Cortex®-R82 processor does not use TTBR.CnP and the TLB is private to a core.

C.3 CONSTRAINED UNPREDICTABLE behavior due to inadequate context synchronization

This section describes the specification and implementation of context synchronization.

Specification

The Arm architecture requires that changes to System registers must be synchronized before they take effect. This can lead to **UNPREDICTABLE** behavior if the synchronization has not been performed. Where multiple control values are updated but not yet synchronized, each control value might independently be the old value or the new value.

Implementation

For the Cortex®-R82 processor, control bits in System registers that require explicit synchronization to reflect the updated value are listed below. All other System register control bits are updated immediately and do not require explicit context synchronization.

- SCTLR_EL2 bits: EE, WXN, BR, I, NAA, C, A, M, SA, ENDA, ENDB, ENIA, ENIB
- SCTLR_EL1 bits: EE, WXN, EOE, BR, I, NAA, SA0, SA, C, A, M, ENDA, ENDB, ENIA, ENIB
- IMP_ITCMREGIONR_EL1 bits: EL2, EL10
- IMP_DTCMREGIONR_EL1 bits: EL2, EL10
- IMP_LLPPREGIONR_EL1 bits: EL2, EL10
- IMP_SPPREGIONR_EL1 bits: EL2, EL10
- IMP_LLDRAMREGIONR_EL1 bits: EL2, EL10
- IMP_MEMPROTCTLR_EL1 bit: MEMPROTEN
- HCR_EL2 bits: AMO, FMO, PTW, IMO, TGE, ID, DC, CD, VM, FWB, HCD
- CPACR_EL1 bits: All
- CPTR_EL2 bits: All
- CPUACTLR_EL1 bits: IALLOC and IMCWT
- DBGAUTHSTATUS_EL1 bits: DBGEN, SPIDEN
- EDSCR bits: HDE, TDA, TFO, INTdis
- OSLSR_EL1 bits: OSLK

C.4 Translation table base address alignment

This section describes the specification and implementation of translation table base address alignment.

Specification

In the translation table base registers TTBRO_EL1, TTBR1_EL1, register bits[48:x] hold the translation table base address, where x depends on the translation table granule size and the

size of the addressed translation table. Register bits[(x-1):0], correspond to bits[(x-1):0] of the translation table base address and therefore are RES0.

For these registers, if one or more RES0 bits in register bits [(x-1):0] does not have a value of 0, this can result in a misaligned translation table base address. In this case, one of the following behaviors must occur:

- The field that is defined to be RES0 is treated as if all the bits had a value of 0:
 - The value read back might be the value written or it might be zero.
- The calculation of an address for a translation table walk using those registers might be corrupted in those bits that are nonzero.

Implementation

For Cortex®-R82 processor, if one or more RES0 bits in register bits [(x-1):0] does not have a value of 0, the field that is defined to be RES0 is treated as if all the bits had a value of 0 and the value read back is the value written.

C.5 The Performance Monitors Extension

This section describes the specification and implementation of accessing the Performance Monitors Extension.

C.5.1 CONSTRAINED UNPREDICTABLE accesses to PMXEVTYPER_ELO or PMXEVCNTR_ELO

This section describes the specification and implementation of accessing to PMXEVTYPER_ELO or PMXEVCNTR_ELO.

Specification

If PMSELR_ELO.SEL is greater than the number of event counters accessible at this Exception level, accesses to PMXEVTYPER_ELO and PMXEVCNTR_ELO can cause **CONSTRAINED UNPREDICTABLE** behavior.

Implementation

In the Cortex®-R82 processor, accesses to PMXEVTYPER_ELO or PMXEVCNTR_ELO from that state behave as RAZ/WI.

C.5.2 CONSTRAINED UNPREDICTABLE accesses to PMEVCNTR<n>_ELO and PMEVTYPER<n>_ELO

This section describes the specification and implementation of accessing to PMEVCNTR<n>_ELO and PMEVTYPER<n>_ELO.

Specification

If <n> is greater than the number of counters available in the current Exception level and state, reads and writes of PMEVCNTR<n>_ELO and PMEVTYPER<n>_ELO are **CONSTRAINED UNPREDICTABLE**.

Implementation

For the Cortex®-R82 processor, <n> is between 0-5 and accesses to the register are **UNDEFINED** when <n> is greater than 5.

C.5.3 CONSTRAINED UNPREDICTABLE behavior caused by MDCR_EL2.HPMN

This section describes the specification and implementation of **CONSTRAINED UNPREDICTABLE** behavior caused by MDCR_EL2.HPMN.

Specification

If MDCR_EL2.HPMN is set to 0, or to a value larger than PMCR_ELO.N, then the value returned by a direct read of MDCR_EL2.HPMN is **UNKNOWN**.

Implementation

If MDCR_EL2.HPMN is set to 0 or to a value larger than PMCR_ELO.N, then an **UNKNOWN** number of counters are reserved for EL2 use. That is, the PE behaves as if MDCR_EL2.HPMN is set to an **UNKNOWN** non-zero value less than or equal to PMCR_ELO.N.

C.6 The Activity Monitors Extension

This section describes the specification and implementation of the Activity Monitors Extension.

Specification

If <n> is greater than the number of architected activity monitor event counters, reads and writes of AMEVCNTR0<n>_ELO and AMEVTYPER0<n>_ELO are **CONSTRAINED UNPREDICTABLE**.

Implementation

The Activity Monitors Extension is not implemented in the Cortex®-R82 processor.

C.7 Syndrome register handling for **CONSTRAINED UNPREDICTABLE** instructions treated as **UNDEFINED**

This section describes the specification and implementation of the syndrome register handling.

Specification

When a **CONSTRAINED UNPREDICTABLE** instruction is treated as **UNDEFINED**, ESR_ELx is **UNKNOWN**.

Implementation

CONSTRAINED UNPREDICTABLE instructions are not treated as **UNDEFINED** in the Cortex®-R82 processor.

C.8 Out of range virtual address

This section describes the specification and implementation of the out of range virtual address.

Specification

If the PE executes a load or store instruction with tagged addressing disabled in the current translation regime, and where the computed virtual address, total access size, and alignment mean that it accesses the bytes at `0xFFFFFFFFFFFFFFFF` and `0x0000000000000000`, then the bytes that appear to be from `0x0000000000000000` onwards are accessed at an **UNKNOWN** address.

If the PE executes a load or store instruction with tagged addressing enabled in the current translation regime, and where the computed address, total access size, and alignment mean that it accesses the bytes at `0xFFFFFFFFFFFFFFFF` and `0x0000000000000000`, then the bytes that appear to be from `0x0000000000000000` onwards are accessed at an **UNKNOWN** address and the tags associated with address also become **UNKNOWN**.

Implementation

Accesses to out of range virtual addresses cause an abort in the Cortex®-R82 processor.

C.9 Mapping of non-idempotent memory locations using the Normal memory type

This section describes the specification and implementation of mapping of non-idempotent memory locations using the Normal memory type.

Specification

If non-idempotent memory locations are mapped using the Normal memory type, the state of the non-idempotent-memory location may become corrupted in following circumstances:

- Speculative read accesses may cause accesses to the non-idempotent memory locations that would not occur as part of a simple sequential execution.
- Writes to non-idempotent memory locations might be merged or split. In this case, the number and size of writes seen by the memory location might not be the number and size that occur as part of a simple sequential execution.

Implementation

The Cortex®-R82 processor implementation is the same as the specification.

C.10 Instruction fetches from Device memory

This section describes the specification and implementation of instruction fetches from Device memory.

Specification

Instruction fetches from Device memory are **CONSTRAINED UNPREDICTABLE**.

Implementation

If a location in memory has the Device attribute and is not marked as execute-never, then the Cortex®-R82 processor will report a permission fault.

C.11 Programming the CSSELR_EL1.Level for a cache level that is not implemented

This section describes the specification and implementation of programming the CSSELR_EL1.Level for a cache level that is not implemented.

Specification

If the CSSELR_EL1.Level is programmed to a cache level that is not implemented, then a read of CSSELR_EL1 returns an **UNKNOWN** value in CSSELR_EL1.Level.

If CSSELR_EL1.Level is programmed to a cache level that is not implemented, then on a read of CCSIDR_EL1 an implementation must perform one of the following behaviors:

- The CCSIDR_EL1 read is treated as a NOP.
- The CCSIDR_EL1 read is **UNDEFINED**.
- The CCSIDR_EL1 read returns an **UNKNOWN** value.

Implementation

If the CSSELR_EL1.Level is programmed to a cache level that is not implemented, then a read of CSSELR_EL1 returns an **UNKNOWN** value in CSSELR_EL1.Level.

If CSSELR_EL1.Level is programmed to a cache level that is not implemented, then a read of CCSIDR_EL1 returns an **UNKNOWN** value.

C.12 Crossing a page boundary with different memory types or Shareability attributes

This section describes the specification and implementation of crossing a page boundary with different memory types or Shareability attributes.

Specification

A memory access from a load or store instruction that crosses a page boundary to a memory location that has a different memory type or Shareability attribute results in **CONSTRAINED UNPREDICTABLE** behavior.

Implementation

The Cortex®-R82 processor has the following behavior for crossing a page (or translation) boundary with different memory types or Shareability attributes:

- Each memory access generated by the instruction uses the memory type and Shareability attribute associated with its own address.
- For alignment checking, the first address of the instruction is used. Therefore, if crossing from Normal to Device memory, an alignment fault would be taken on the second access if not aligned to element size. The Cortex®-R82 processor treats any accesses that cross a 128-bit boundary as independent accesses, so all alignment, permission, and other fault checking is performed for each aligned 128-bit access. This means that if an access is permitted to the lower 128-bit aligned region but not to the higher 128-bit aligned region, then the first access will go ahead. This may mean that the lower addresses are updated if there is a store that gets a fault on the second access.

C.13 CONSTRAINED UNPREDICTABLE behaviors with Load-Exclusive/Store-Exclusive pairs

This section describes the specification and implementation of Load-Exclusive/Store-Exclusive pairs.

Load-Exclusive and Store-Exclusive instruction usage restrictions in the Arm®v8 architecture defines a Load-Exclusive/Store-Exclusive pair, and identifies various **CONSTRAINED UNPREDICTABLE** behaviors associated with using Load-Exclusive/Store-Exclusive pairs. These cases and their implementation in the Cortex®-R82 processor are:

- The target virtual address of a StoreExcl instruction is different from the virtual address of the preceding LoadExcl instruction in the same thread of execution.

Implementation

This will cause the StoreExcl to fail, the status value returned by the StoreExcl is **UNKNOWN**, and the states of the local and global monitors for that PE are **UNKNOWN**. The data at the address accessed by the LoadExcl, and at the address accessed by the StoreExcl, is **UNKNOWN**.

- The transaction size of a StoreExcl instruction is different from the transaction size of the preceding LoadExcl instruction in the same thread of execution.

Implementation

The Cortex®-R82 processor does not require the transaction size to be the same.

- The StoreExcl instruction accesses a different number of registers than the preceding LoadExcl instruction in the same thread of execution.

Implementation

The Cortex®-R82 processor does not require that StoreExcl instruction access the same number of registers as the preceding LoadExcl instruction. Therefore a difference in the number of registers accessed does not affect whether a StoreExcl passes or fails.

- The memory attributes for a StoreExcl instruction are different from the memory attributes for the preceding LoadExcl instruction in the same thread of execution.

Implementation

If Cacheability or Shareability differs between the LoadExcl and the StoreExcl, the store exclusive will fail.

- The effect of a data or unified cache invalidate, clean, or clean and invalidate instruction on a local or global Exclusives monitor that is in the Exclusive Access state is **CONSTRAINED UNPREDICTABLE**.

Implementation

If the load exclusive is allocated into the cache and has shareable attributes, cache invalidation will cause the monitor to be opened.

C.14 CONSTRAINED UNPREDICTABLE behavior for instructions

This section describes **CONSTRAINED UNPREDICTABLE** behavior for instructions.

C.14.1 LDAXP, LDNP, LDNP (SIMD&FP)

This section describes the specification and implementation of LDAXP, LDNP, LDNP (SIMD&FP).

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If $t == t2$, then the instruction performs a load using the specified addressing mode, and the transfer register is set to an **UNKNOWN** value.

C.14.2 LDP

This section describes the specification and implementation of LDP.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If the instruction encoding specifies pre-indexed addressing or post-indexed addressing, and $(t == n \parallel t2 == n) \&\& n \neq 31$, then the instruction performs a load using the specified addressing mode, and the base register is set to an **UNKNOWN** value. In addition, if an exception occurs during such an instruction, the base register might be corrupted so that the instruction cannot be repeated.

If $t == t2$, then the instruction performs all of the loads using the specified addressing mode, and the transfer register is set to an **UNKNOWN** value.

C.14.3 LDP (SIMD&FP)

This section describes the specification and implementation of LDP (SIMD&FP).

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If $t == t2$, then the instruction performs all of the loads using the specified addressing mode, and the transfer register is set to an **UNKNOWN** value.

C.14.4 LDPSW

This section describes the specification and implementation of LDPSW.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If the instruction encoding specifies pre-indexed addressing or post-indexed addressing, and $(t == n \parallel t2 == n) \&\& n \neq 31$, then the instruction performs a load using the specified addressing mode, and the base register is set to an **UNKNOWN** value. In addition, if an exception occurs during such an instruction, the base register might be corrupted so that the instruction cannot be repeated.

If $t == t2$, then the instruction performs all of the loads using the specified addressing mode, and the transfer register is set to an **UNKNOWN** value.

C.14.5 LDR (immediate), LDRB (immediate), LDRH (immediate), LDRSB (immediate), LDRSH (immediate), LDRSW (immediate)

This section describes the specification and implementation of LDR (immediate), LDRB (immediate), LDRH (immediate), LDRSB (immediate), LDRSH (immediate), LDRSW (immediate).

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If the instruction encoding specifies pre-indexed addressing or post-indexed addressing, and $n == t$ && $n != 31$, then the instruction performs the load using the specified addressing mode, and the base register is set to an **UNKNOWN** value. In addition, if an exception occurs during such an instruction, the base register might be corrupted so that the instruction cannot be repeated.

C.14.6 LDXP

This section describes the specification and implementation of LDXP.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If $t == t2$, then the instruction performs a load using the specified addressing mode, and the transfer register is set to an **UNKNOWN** value.

C.14.7 STP

This section describes the specification and implementation of STP.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If the instruction encoding specifies pre-indexed addressing or post-indexed addressing, and $(t == n || t2 == n) \&\& n != 31$, then the instruction performs a store using the specified addressing mode but the value stored is **UNKNOWN**.

C.14.8 STLXP

This section describes the specification and implementation of STLXP.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If $s == t \parallel (s == t2)$, then the instruction performs the store to the specified address, but the value stored is **UNKNOWN**.

If $s == n \ \&\& \ n \neq 31$, then the instruction performs the store to an **UNKNOWN** address.

C.14.9 STLXR, STLXRB, STLXRH

This section describes the specification and implementation of STLXR, STLXRB, STLXRH.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If $s == t$, then the instruction performs the store to the specified address, but the value stored is **UNKNOWN**.

If $s == n \ \&\& \ n \neq 31$ then the instruction performs the store to an **UNKNOWN** address.

C.14.10 STR (immediate), STRB (immediate), STRH (immediate)

This section describes the specification and implementation of STR (immediate), STRB (immediate), STRH (immediate).

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If the instruction encoding specifies pre-indexed addressing or post-indexed addressing, and $n == t \ \&\& \ n \neq 31$, then the instruction performs a store using the specified addressing mode but the value stored is **UNKNOWN**.

C.14.11 STXP

This section describes the specification and implementation of STXP.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If $s == t \parallel (s == t2)$, then the instruction performs the store to the specified address, but the value stored is **UNKNOWN**.

If $s == n \ \&\& \ n \neq 31$ then the instruction performs the store to an **UNKNOWN** address.

C.14.12 STXR, STXRB, STXRH

This section describes the specification and implementation of STXR, STXRB, STXRH.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If $s == t$, then the instruction performs the store to the specified address, but the value stored is **UNKNOWN**.

If $s == n \ \&\& \ n \neq 31$ then the instruction performs the store to an **UNKNOWN** address.

C.15 Out of range values of the Set/Way/Index fields in cache maintenance instructions

This section describes the specification and implementation of Out of range values of the Set/Way/Index fields in cache maintenance instructions.

Specification

In the cache maintenance by set/way instructions `DC C1SW`, `DC C2SW`, and `DC ISW`, if any set/way/index argument is larger than the value supported by the implementation, then the behavior is **CONSTRAINED UNPREDICTABLE**.

Implementation

In this situation, the instruction performs cache maintenance on a single arbitrary cache line.

C.16 Reserved values in System and memory-mapped registers and translation table entries

Specification

Unless otherwise stated in the [Arm® Architecture Reference Manual for A-profile architecture](#), all unallocated or reserved values of fields with allocated values within AArch64 System registers, memory-mapped registers, and translation table entries behave in one of the following ways:

- The unallocated value maps onto any of the allocated values, but otherwise does not cause **CONSTRAINED UNPREDICTABLE** behavior.
- The unallocated value causes effects that could be achieved by a combination of more than one of the allocated values.
- The unallocated value causes the field to have no functional effect.

Implementation

All unallocated or reserved values of fields with allocated values within AArch64 System registers, memory-mapped registers, and translation table entries have no functional effect.

C.17 CONSTRAINED UNPREDICTABLE behavior in Debug state

This section describes the **CONSTRAINED UNPREDICTABLE** behaviors that are specifically associated with Debug state.

C.17.1 Instructions that are CONSTRAINED UNPREDICTABLE in Debug state

This section lists instructions that are **CONSTRAINED UNPREDICTABLE** in Debug state and their implementation behavior.

- Exception-generating instructions
These instructions are: SVC, HVC, BRK, HLT.

Implementation

They are **UNDEFINED**.

- Instructions that explicitly write to the PC
These instructions are: B, B.cond, BL, BLR, BR, CBZ, CBNZ, RET, TBZ, TBNZ.

Implementation

They are **UNDEFINED**.

- Exception return ERET.

Implementation

They are **UNDEFINED**.

- Instructions that request entry to low-power state
These instructions are: WFE, WFI.

Implementation

They are **UNDEFINED**.

- Instructions that read the PC
These instructions are: LDR (literal), LDRSW (literal), ADR, ADRP, PRFM (literal).

Implementation

They are **UNDEFINED**.

- Instructions that explicitly modify PSTATE
These instructions are:
 - ADDS, SUBS, ADCS, SBCS, ANDS, BICS, CCMN, CCMP
 - FCMP, FCMPE, FCCMP, FCCMPE
 - MSR DAIFSet (immediate), MSR DAIFClr (immediate), MSR SPSEL (immediate)
 - MSR NZCV (register), MSR DAIF (register), MSR SPSEL (register)
 - MSR PAN (immediate) and MSR PAN (register)
 - MSR UAO (immediate) and MSR UAO (register)
 - CFINV, RMIF, SETF8, SETF16
 - MSR DIT

Implementation

They are **UNDEFINED**.

- Instructions that read PSTATE.{N, Z, C, V} or other PSTATE fields
These instructions are:
 - CSEL, CSINC, CSINV, CSNEG, CCMN, CCMP, FCSEL, FCCMP, FCCMPE
 - ADC, ADCS, SBC, SBCS
 - CFIINV
 - MRS NZCV, MRS DAIF, MRS SPSEL, MRS CurrentEL
 - MSR PAN
 - MSR UAO
 - MSR DIT

Implementation

They are **UNDEFINED**.

- Hint instruction DGH.

Implementation

They execute as in Non-debug state.

- All other instructions that are not specified either as changed or unchanged in Debug state and are not listed above

Implementation

They are **UNDEFINED**.

C.17.2 Exiting Debug state

This section describes the specification and implementation of exiting Debug state.

Specification

The PE exits Debug state when it receives a Restart request trigger event. If EDSCR.ITE == 0 the behavior of any instruction issued through the ITR in Normal access mode or an operation issued by a DTR access in memory access mode that has not completed execution is **CONSTRAINED UNPREDICTABLE**.

Implementation

In this case the Cortex®-R82 processor completes execution in Debug state before the PE executes the restart sequence.

C.17.3 Changing the value of EDECR.SS when not in Debug state

This section describes the specification and implementation of changing the value of EDECR.SS when not in Debug state.

Specification

If software changes the value of EDECR.SS when the PE is not in Debug state then behavior is **CONSTRAINED UNPREDICTABLE**.

Implementation

In this situation in the Cortex®-R82 processor, the value of EDECR.SS becomes **UNKNOWN**.

C.17.4 Syndrome information on Halting Step

This section describes the specification and implementation of Syndrome information on Halting Step.

EDSCR.STATUS is **CONSTRAINED UNPREDICTABLE** when:

- The instruction being stepped generated a Halting Step debug event before the instruction was executed.

Implementation

In this case EDSCR.STATUS is set to:

- Halting Step, no syndrome, if the stepped instruction was not a Load-Exclusive instruction
- Halting Step, no syndrome, if the stepped instruction was a Load-Exclusive instruction

- The instruction that was stepped was an Exception Return instruction or an ISB.

Implementation

In this case EDSCR.STATUS is set to Halting Step, no syndrome.

C.17.5 Illegal Execution state exception

This section describes the specification and implementation of Illegal Execution state exception.

Specification

If PSTATE.IL is set to 1 when EDSCR.MA == 1, then on an external write access to DBGDTRRX_ELO or an external read from DBGDTRTX_ELO, it is **CONSTRAINED UNPREDICTABLE**.

Implementation

The Cortex®-R82 processor ignores PSTATE.IL.

C.17.6 Alignment constraints

This section describes the specification and implementation of alignment constraints.

Specification

See *CONSTRAINED UNPREDICTABLE behavior for A64 instructions* chapter of the [Arm® Architecture Reference Manual for A-profile architecture](#).

Implementation

If the address in R0 is not aligned to a multiple of four, the Cortex®-R82 processor makes an unaligned memory access to R0. If alignment checking is enabled for the memory access, this generates an Alignment fault.

C.17.7 Cumulative error flag

This section describes the specification and implementation of cumulative error flag.

Specification

The cumulative error flag remains set until cleared to 0 by writing 1 to EDRCCR.CSE. However, the effect of writing 1 to EDRCCR.CSE to clear EDSCR.ERR is **CONSTRAINED UNPREDICTABLE** when both of the following apply:

- The PE is in Debug state
- The value of EDSCR.ITE is 0

Implementation

When these conditions apply and a value of 1 is written to EDRCCR.CSE, the Cortex®-R82 processor clears the cumulative error flag EDSCR.ERR.

C.17.8 Restart request trigger event

This section describes the specification and implementation of Restart request trigger event.

Specification

If a Restart request trigger event is received at or about the same time as the PE enters Debug state, it is **CONSTRAINED UNPREDICTABLE** whether:

- The request is ignored by the PE. In this case the PE enters Debug state and remains in Debug state.
- The PE enters Debug state and then immediately restarts.

Implementation

The Cortex®-R82 processor enters Debug state and then immediately restarts.

C.17.9 External debug interface accesses to registers in reset

This section describes the specification and implementation of external debug interface accesses to registers in reset.

Specification

If a reset signal is asserted and the external debug interface writes a register, or indirectly writes a register or register field as a side-effect of an access:

- Then, if the register or register field is reset by that reset signal, it is **CONSTRAINED UNPREDICTABLE** whether the register or register field takes the reset value or the value written. The reset value might be **UNKNOWN**.
- Otherwise, the register or register field takes the value that is written.

Implementation

In this case in the Cortex®-R82 processor, the register or register field takes the reset value.

C.17.10 Reserved and unallocated registers

This section describes the specification and implementation of reserved and unallocated registers.

Specification

See [Arm® Architecture Reference Manual for A-profile architecture](#) for information on reserved and unallocated registers.

Implementation

For reserved Debug registers and Performance Monitors registers, if the core power domain is off state, the response is an Error. If the core is in retention, these accesses initiate a wakeup and the response is not an Error.

C.17.11 External accesses to DBGBVR<n>_EL1 and DBGBCR<n>_EL1

This section describes the specification and implementation of external accessing to DBGBVR<n>_EL1 and DBGBCR<n>_EL1.

Specification

See [Arm® Architecture Reference Manual for A-profile architecture](#) for information on external accesses to DBGBVR<n>_EL1 and DBGBCR<n>_EL1.

Implementation

If breakpoint n is not implemented then accesses to these registers return an Error if either core is off, or `os_Lock` is set, or `AllowExternalDebugAccess()` is false.

C.17.12 External accesses to DBGWVR<n>_EL1 and DBGWCR<n>_EL1

This section describes the specification and implementation of external accessing to DBGWVR<n>_EL1 and DBGWCR<n>_EL1.

Specification

See [Arm® Architecture Reference Manual for A-profile architecture](#) for information on external accesses to DBGWVR<n>_EL1 and DBGWCR<n>_EL1.

Implementation

If watchpoint n is not implemented then accesses to these registers return an Error if either core is off, or `os_Lock` is set, or `AllowExternalDebugAccess()` is false.

C.17.13 Accessing the EDESR

This section describes the specification and implementation of accessing the EDESR.

Specification

If a request to clear a pending Halting debug event is received at or about the time when halting becomes allowed, it is **CONSTRAINED UNPREDICTABLE** whether the event is taken.

Implementation

In this case, the event is not taken.

C.17.14 Accessing the CTIAPPPULSE

This section describes the specification and implementation of accessing the CTIAPPPULSE.

Specification

It is **CONSTRAINED UNPREDICTABLE** whether a write to CTIAPPPULSE generates an event on a channel if `CTICONTROL.GLBEN` is 0.

Implementation

In this case, the Cortex®-R82 processor will not generate an output channel event

C.18 RAS registers

In ERR<n>MISC0, Error Record Miscellaneous Register 0, the Corrected Error Counter CEC[38:32] counts only Corrected errors.

Deferred and Uncorrected errors are not counted.

See [B.1.2.1.4 ERR<n>MISC0, Error Record <n> Miscellaneous Register 0, n = 0 - 9](#) on page 1436 for more information.

Appendix D Generic handler example

This appendix contains example interrupt handling sequences.

D.1 Generic handler example, part 1

An example generic handler preamble code snippet is provided here, under the following assumptions:

- No branch required from the exception vector address, because the generic interrupt handler preamble can wholly fit (it is less than 32 instructions, or 0x80 bytes); if a bigger preamble is needed, a branch instruction can be added.
- Re-entrant interrupts are desirable; if not, the SPSR/ELR saving and the DAIFCLR access can be omitted.
- Non-generic interrupt handler does not use FP/NEON registers; if it needs to, the caller-saved FP/NEON registers must be saved as well.
- The handler-specific stack is used; if the thread-specific stack is needed instead, an `MSR SPSEL` instruction must be added.

This example shows the preamble of an EL1 interrupt handler for IRQ. A similar code snippet could be applicable for EL2 and/or FIQ or asynchronous aborts. The code has been optimized to take advantage of the superscalar capabilities in the Cortex®-R82 processor, for example, instructions have been organized to allow for maximal multi-issuing.

```
// Save the first batch of the caller-saved registers to the stack.
STP X1, X2, [SP, #-16]!
STP X0, X3, [SP, #-16]!

// Read the ID of the highest-priority interrupt in X0,
// and acknowledge the interrupt.
// Read the exception context in X1 and X2.
// Meanwhile, continue saving caller-saved registers.
MRS X0, ICC_IAR0_EL1
STP X4, X5, [SP, #-16]!
MRS X1, SPSR_EL1
STP X6, X7, [SP, #-16]!
MRS X2, ELR_EL1
STP X8, X9, [SP, #-16]!

// Re-enable IRQ interrupts. Higher-priority interrupts can now be taken.
MSR DAIFCLR, #2

// Save the rest of the caller-saved registers.
// We have already done X0-X9, but we are saving again X0-X2 that contain
// the interrupt ID and the exception context. We will need them again after
// the non-generic handler returns, to end the interrupt and
// to perform the exception return.
//
// Meanwhile, find the correct non-generic interrupt handler function
// for this IRQ by looking up the interrupt ID in a table of handlers.
// In C notation, the computed function address is X3 = irq_table_base[X0 * 8].
STP X10, X11, [SP, #-16]!
STP X12, X13, [SP, #-16]!
ADR X3, irq_table_base
```

```

STP X14, X15, [SP, #-16]!
ADD X3, X3, X0, LSL #3
STP X16, X17, [SP, #-16]!
STP X29, X30, [SP, #-16]!
LDR X3, [X3]
STP X1, X2, [SP, #-16]!
STP X0, X18, [SP, #-16]!

// Branch to non-generic handler.
// Interrupt ID is passed as the first argument in X0.
// SPSR and ELR are passed as arguments in X1 and X2 as well.
BLR X3

// Branch to epilogue code, if it cannot fit in the exception vector.
B generic_handler_epilogue

```

D.2 Generic handler example, part 2

After the non-generic handler function returns, an example generic handler epilogue is provided in the following code snippet. The example continues the EL1 IRQ handler example with similar assumptions. This example epilogue cannot fit together with the example preamble in the exception vector address, as their combined size is slightly more than 32 instructions. For this reason, a branch is assumed to be required after the non-generic handler return to point to the epilogue code that is in another memory section.

```

// Restore the interrupt ID (X0) and the exception context (X1, X2).
// Also restore the caller-saved X18, to take advantage of load-pair instructions.
generic_handler_epilogue:
LDP X0, X18, [SP], #16
LDP X1, X2, [SP], #16

// Tell the GIC that this interrupt ID has been handled, and it can be deactivated.
// Note that if the source of the interrupt has to be cleared,
// this has already been taken care of in the non-generic interrupt handler.
MSR ICC_EOIR0_EL1, X0

// Continue restoring caller-saved registers.
LDP X29, X30, [SP], #16
LDP X16, X17, [SP], #16
LDP X14, X15, [SP], #16
LDP X12, X13, [SP], #16
LDP X10, X11, [SP], #16
LDP X8, X9, [SP], #16
LDP X6, X7, [SP], #16
LDP X4, X5, [SP], #16
LDP X0, X3, [SP], #16

// Restore the SPSR and ELR so that we can return from the exception.
// Disable interrupts before doing this, to avoid another interrupt corrupting them.
MSR DAIFSet, #2
MSR SPSR_EL1, X1
MSR ELR_EL1, X2

// Restore remaining caller-saved registers.
LDP X1, X2, [SP], #16

// Return from the exception.
ERET

```

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

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

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted 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 this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in the Arm documents.

Product status

All products and Services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Product revision status

This product is r3p1, which indicates the revision status of the product described in this manual, where:

- r (value)**Identifies the major revision of the product, for example, r1.
- p (value)**Identifies the minor revision or modification status of the product, for example, p2.

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
0301-05	25 October 2024	Non-Confidential	First release for r3p1
0300-04	19 March 2024	Non-Confidential	First early access release for r3p0
0200-03	27 October 2023	Non-Confidential	First early access release for r2p0
0101-02	24 November 2022	Non-Confidential	First early access release for r1p1
0100-01	14 July 2022	Confidential	First limited access release for r1p0

The Change history tables describe the technical changes between released issues of this document in reverse order. Issue numbers match the revision history in [Document release information](#) on page 2201.

Table 2: Issue 0100-01

Change	Location
First limited access release for r1p0	-

Table 3: Differences between issue 0100-01 and issue 0101-02

Change	Location
First early access release for r1p1	-
Updated the Figure Cortex®-R82 cluster PPU mode transitions	5.11 Cluster PPU mode transitions on page 73
Updated the values for Half L2 cache in Table Operating mode enumeration for the cluster	6.4 Encodings for cluster power modes and operating modes on page 84
Added a row for Max_Transaction_Bytes in Table LLRAM features	8.8.1 LLRAM features on page 157
Removed the LLRAM region from the cases that generate an OK response on RRESPS or BRESPS	8.10.3.1 ACELS transaction restrictions on page 171
Corrected External PPU_IIDR and PPU_PIDR<n> register descriptions to reflect compatibility to PCK-600 Power Policy Unit	B.1.1.2 External PPU registers summary on page 1417
Corrected External CLUSTERPPU_IIDR and CLUSTERPPU_PIDR<n> register descriptions to reflect compatibility to PCK-600 Power Policy Unit	B.1.1.3 External CLUSTERPPU registers summary on page 1419
Terminology changes per Arm's Inclusive language commitment: <ul style="list-style-type: none"> For AXI interface: Used <i>Manager</i> to indicate the agent that initiates transactions and <i>Subordinate</i> to indicate the agent that receives and responds to requests. For AXI-Stream interface: Used <i>Transmitter</i> to indicate the agent that initiates transactions and <i>Receiver</i> to indicate the agent that receives and responds to requests. For APB interface: Used <i>Requester</i> to indicate the agent that initiates transactions and <i>Completer</i> to indicate the agent that receives and responds to requests. For ATB interface: Used <i>Transmitter</i> to indicate the agent that initiates transactions and <i>Receiver</i> to indicate the agent that receives and responds to requests. 	Throughout manual

Table 4: Differences between issue 0101-02 and issue 0200-03

Change	Location
First early access release for r2p0	-
Added AMBA 5 ATB protocol (Issue C) to the Bus architecture specification list	1.4 Supported standards and specifications on page 23
Updated the Processor interfaces figure to indicate the inclusion of CHI.e features across the processor	2.3 Interfaces on page 35
Updated the Processor interfaces table to detail the inclusion of CHI.e as an option for the Main Manager	2.3 Interfaces on page 35
Updated the Architecture requirements table to reflect the architectural features required by this revision	3.2.1 Architecture requirements on page 39
Updated the Full Retention (FULL_RET) information of the Core Power Modes and Transitions to detail the transition between full power and on modes	5.7 Core power modes and transitions on page 62
Updated the Memory retention (MEM_RET) information of the Core Power Modes and Transitions to detail the transition between Memory retention and on modes	5.9 Cluster power modes on page 68

Change	Location
Updated the Main Master Interface chapter globally to reflect the addition of CHI features	8.7 MM interface on page 140
Updated the LLRAM manager interface chapter to reflect the addition of Cross-cluster LLRAM sharing when CFGLLRAMSHARED=1	8.8 LLRAM manager interface on page 155

Table 5: Differences between issue 0200-03 and issue 0300-04

Change	Location
First early access release for r3p0	-
Updated Physical Address Width (PA_W) to allow configurations to be set to either 40 or 48	Throughout manual and 1.3.1 Configuration parameters on page 15

Table 6: Differences between issue 0300-04 and issue 0301-05

Change	Location
First release for r3p1	-
Front matter and back matter order change	A change was made to the ordering of the front matter and back matter.
Architectural registers were updated with relevant changes for this release.	A. AArch64 registers on page 279
External registers were updated with relevant changes for this release.	B. External registers on page 1415
Memory Systems Chapter was updated with release correct specifications in every table throughout.	8. Memory system on page 103
PMU events were updated with release correct specifications.	14. PMU on page 249
Processor features chapter was revised to ensure clarity of text.	2. Technical overview on page 28

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

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

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

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
<i>italic</i>	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 <u>underline</u>	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

Convention	Use
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <div>MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2></div>
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 information is important and needs your attention.



This information might help you perform a task in an easier, better, or faster way.



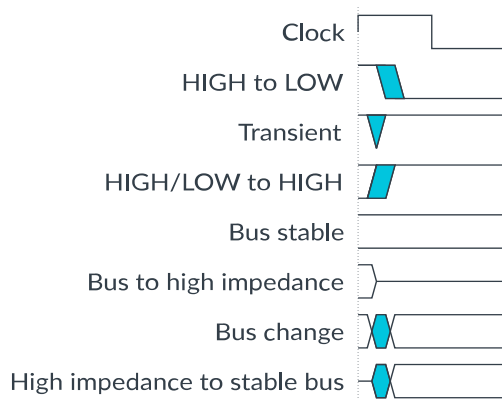
This information reminds you of something important relating to the current content.

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: 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.

Useful resources

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

Access to Arm documents depends on their confidentiality:

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

Arm product resources	Document ID	Confidentiality
Arm® Cortex®-R82 Processor Configuration and Integration Manual	102671	Confidential
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101088	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
AMBA® APB Protocol Specification	IHI 0024	Non-Confidential
AMBA® AXI Protocol Specification	IHI 0022	Non-Confidential
AMBA® AXI-Stream Protocol Specification	IHI 0051	Non-Confidential
AMBA® Low Power Interface Specification	IHI 0068	Non-Confidential
Arm® Architecture Reference Manual Supplement Armv8, for R-profile AArch64 architecture	DDI 0600	Non-Confidential
Arm® AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1	IHI 0032	Non-Confidential
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	Non-Confidential
Arm® Embedded Trace Macrocell Architecture Specification ETMv4.0 to ETM4.6	IHI 0064	Non-Confidential
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	Non-Confidential
Arm® Power Policy Unit Architecture Specification	DEN 0051	Non-Confidential
Arm® Architecture Reference Manual Supplement Reliability, Availability, and Serviceability (RAS), for A-profile architecture	DDI 0587	Non-Confidential