Arm® Cortex-A78C Core

Revision: r0p1

Technical Reference Manual

arm

102226_0001_01_en

Arm® Cortex-A78C Core

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0001-01	02 November 2020	Non-Confidential	First early access release for r0p1

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

developer.arm.com

Contents

Arm® Cortex-A78C Core Technical Reference Manual

Preface

This preface introduces the *Arm® Cortex-A78C Core Technical Reference Manual*.

It contains the following:

About this book

This Technical Reference Manual is for the Cortex-A78C core. It provides reference documentation and contains programming details for registers. It also describes the memory system, the caches, the interrupts, and the debug features.

Product revision status

The r*x*p*y* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

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

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

Intended audience

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

Using this book

This book is organized into the following chapters:

Part A Functional description
This part describes the main functionality of the Cortex-A78C core.

Chapter A1 Introduction
This chapter provides an overview of the Cortex-A78C core and its features.

Chapter A2 Technical overview
This chapter describes the structure of the Cortex-A78C core.

Chapter A3 Clocks, resets, and input synchronization
This chapter describes the clocks, resets, and input synchronization of the Cortex-A78C core.

Chapter A4 Power management
This chapter describes the power domains and the power modes in the Cortex-A78C core.

Chapter A5 Memory Management Unit
This chapter describes the *Memory Management Unit* (MMU) of the Cortex-A78C core.

Chapter A6 L1 memory system
This chapter describes the L1 instruction cache and data cache that make up the L1 memory system.

Chapter A7 L2 memory system
This chapter describes the L2 memory system.

Chapter A8 Reliability, Availability, and Serviceability
This chapter describes the *Reliability, Availability, and Serviceability* (RAS) features implemented in the Cortex-A78C core.

Chapter A9 Generic Interrupt Controller CPU interface
This chapter describes the Cortex-A78C core implementation of the Arm *Generic Interrupt Controller* (GIC) CPU interface.

Chapter A10 Advanced SIMD and floating-point support
This chapter describes the Advanced SIMD and floating-point features and registers in the Cortex-A78C core. The unit in charge of handling the Advanced SIMD and floating-point features is also referred to as the data engine in this manual.

Part B Register descriptions
This part describes the non-debug registers of the Cortex-A78C core.

Chapter B1 AArch32 system registers
This chapter describes the system registers in the AArch32 state.

Chapter B2 AArch64 system registers
This chapter describes the system registers in the AArch64 state.

Chapter B3 Error system registers
This chapter describes the error registers accessed by the AArch64 error registers.

Chapter B4 GIC registers
This chapter describes the GIC registers.

Chapter B5 Advanced SIMD and floating-point registers
This chapter describes the Advanced SIMD and floating-point registers.

Part C Debug descriptions
This part describes the debug functionality of the Cortex-A78C core.

Chapter C1 Debug
This chapter describes the Cortex-A78C core debug registers and shows examples of how to use them.

Chapter C2 Performance Monitoring Unit
This chapter describes the *Performance Monitoring Unit* (PMU) and the registers that it uses.

Chapter C3 Activity Monitor Unit
This chapter describes the *Activity Monitor Unit* (AMU).

Chapter C4 Embedded Trace Macrocell
This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex-A78C core.

Chapter C5 Statistical Profiling Extension
This chapter describes the *Statistical Profiling Extension* (SPE) for the Cortex-A78C core.

Part D Debug registers
This part describes the debug registers of the Cortex-A78C core.

Chapter D1 AArch32 debug registers
This chapter describes the debug registers in the AArch32 Execution state and shows examples of how to use them.

Chapter D2 AArch64 debug registers
This chapter describes the debug registers in the AArch64 Execution state and shows examples of how to use them.

Chapter D3 Memory-mapped debug registers
This chapter describes the memory-mapped debug registers and shows examples of how to use them.

Chapter D4 AArch32 PMU registers
This chapter describes the AArch32 *Performance Monitoring Unit* (PMU) registers and shows examples of how to use them.

Chapter D5 AArch64 PMU registers
This chapter describes the AArch64 *Performance Monitoring Unit* (PMU) registers and shows examples of how to use them.

Chapter D6 Memory-mapped PMU registers
This chapter describes the memory-mapped *Performance Monitoring Unit* (PMU) registers and shows examples of how to use them.

Chapter D7 PMU snapshot registers
Performance Monitoring Unit (PMU) snapshot registers are an IMPLEMENTATION DEFINED extension to an Armv8-A compliant PMU to support an external core monitor that connects to a system profiler.

Chapter D8 AArch64 AMU registers
This chapter describes the AArch64 *Activity Monitor Unit* (AMU) registers and shows examples of how to use them.

Chapter D9 Memory-mapped AMU registers
This chapter describes the memory-mapped *Activity Monitor Unit* (AMU) registers. The memory-mapped interface provides read-only access to the AMU registers via the external debug interface.

Chapter D10 ETM registers
This chapter describes the *Embedded Trace Macrocell* (ETM) registers.

Chapter D11 SPE registers
This chapter describes the *Statistical Profiling Extension* (SPE) registers.

Part E Appendices
This part describes the appendices of the Cortex-A78C core.

Appendix A Cortex-A78C Core AArch32 UNPREDICTABLE behaviors
This appendix describes the cases in which the Cortex-A78C core implementation diverges from the preferred behavior described in Armv8 AArch32 UNPREDICTABLE behaviors.

Appendix B Revisions
This appendix describes the technical changes between released issues of this book.

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.

Typographic conventions
italic
Introduces special terminology, denotes cross-references, and citations.

bold
Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`
Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`monospace`
Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`
Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`
Denotes language keywords when used outside example code.

`<and>`
Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

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.

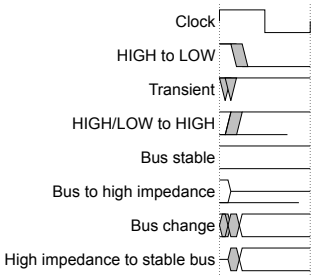

Figure 1 Key to timing diagram conventions

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.

Additional reading

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

Arm publications

- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* (DDI 0487)
- *Arm® Cortex-A78C Core Configuration and Integration Manual* (102227)
- *Arm® Cortex-A78C Core Cryptographic Extension Technical Reference Manual* (102228)
- *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual* (101512)
- *Arm® DynamIQ™ Shared Unit MP135 Configuration and Sign-off Guide* (101513)
- *Arm® DynamIQ™ Shared Unit MP135 Integration Manual* (101514)
- *Arm® CoreSight™ ELA-500 Embedded Logic Analyzer Technical Reference Manual* (100127)
- *AMBA® AXI and ACE Protocol Specification* (IHI 0022)
- *AMBA® APB Protocol Version 2.0 Specification* (IHI 0024)
- *AMBA® 4 ATB Protocol Specification* (IHI 0032)
- *AMBA® 5 CHI Architecture Specification* (IHI 0050)
- *AMBA® Low Power Interface Specification Arm® Q-Channel and P-Channel Interfaces* (IHI 0068)
- *Arm® CoreSight™ Architecture Specification v3.0* (IHI 0029)
- *Arm® Debug Interface Architecture Specification, ADIv5.0 to ADIv5.2* (IHI 0031)
- *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* (IHI 0069)
- *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* (IHI 0064)
- *Arm® Reliability, Availability, and Serviceability (RAS) Specification, Armv8, for the Armv8-A architecture profile* (DDI 0587A)

Other publications

- *ANSI/IEEE Std 754-2008, IEEE Standard for Binary Floating-Point Arithmetic*.

  ———— **Note** ————

  Arm floating-point terminology is largely based on the earlier ANSI/IEEE Std 754-1985 issue of the standard. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

  ————————————

Feedback

Feedback on this product

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

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

Feedback on content

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

- The title *Arm Cortex-A78C Core Technical Reference Manual*.
- The number 102226_0001_01_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

——— **Note** ———

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

———————————

Part A
Functional description

Chapter A1
Introduction

This chapter provides an overview of the Cortex-A78C core and its features.

It contains the following sections:

A1.1 About the core

The Cortex-A78C core is a high-performance and low-power Arm product that implements the Armv8-A architecture.

The Cortex-A78C core supports:

- The Armv8.2-A extension
- The *Statistical Profiling Extension* (SPE)
- The *Reliability, Availability, and Serviceability* (RAS) extension
- The Pointer Authentication introduced in the Armv8.3-A extension
- The Load acquire (`LDAPR`) instructions introduced in the Armv8.3-A extension
- The multi-OS support introduced in the Armv8.4-A extension
- The Dot Product support instructions introduced in the Armv8.4-A extension
- The traps for EL0 and EL1 cache controls, PSTATE *Speculative Store Bypass Safe* (SSBS) bit and the speculation barriers (CSDB, SSBB, PSSBB) instructions introduced in the Armv8.5-A extension
- The Enhanced Pointer Authentication, excluding the optional *Faulting Pointer Authentication Code* (FPAC) extension, introduced in the Armv8.6-A extension

The Cortex-A78C core has an L1 memory system and a private, integrated L2 cache. It also includes a superscalar, variable-length, out-of-order pipeline.

The Cortex-A78C core is implemented inside the *DynamIQ Shared Unit* (DSU) cluster. For more information, see the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

The following figure shows an example of a configuration with four Cortex-A78C cores.

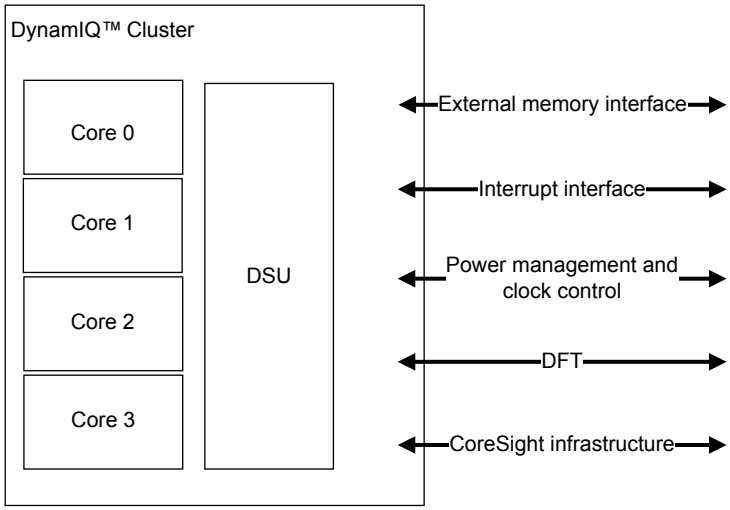

Figure A1-1 Example Cortex-A78C configuration

A1.2 Features

The features of the Cortex-A78C core are categorized as core, cache, or debug features.

Core features

- 40-bit *Physical Address* (PA)
- Optional Cryptographic Extension
- A *Memory Management Unit* (MMU)
- Support for Arm TrustZone® technology
- Armv8.4-A Dot Product instruction support
- Superscalar, variable-length, out-of-order pipeline
- Support for *Page-Based Hardware Attributes* (PBHA)
- *Reliability, Availability, and Serviceability* (RAS) Extension
- Full implementation of the Armv8.2-A A64, A32, and T32 instruction sets
- *Generic Interrupt Controller* (GICv4) CPU interface to connect to an external distributor
- Generic Timers interface supporting 64-bit count input from an external system counter
- AArch32 Execution state at Exception level EL0 only. AArch64 Execution state at all Exception levels (EL0 to EL3)
- An integrated execution unit that implements the Advanced SIMD and floating-point architecture support

Cache features

- Separate L1 data and instruction caches
- Private, unified data and instruction L2 cache
- Optional L1 and L2 memory protection in the form of *Error Correcting Code* (ECC) or parity on all RAM instances which affect functionality

Debug features

- Armv8.2 debug logic
- *Performance Monitoring Unit* (PMU)
- *Statistical Profiling Extension* (SPE)
- Optional *CoreSight Embedded Logic Analyzer* (ELA)
- *Embedded Trace Macrocell* (ETM) that supports instruction trace only

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

Cortex-A78C feature additions

- Armv8.3-A Pointer Authentication support
- Armv8.4-A multi-OS support
- Armv8.6-A Enhanced Pointer Authentication support, excluding the optional FPAC extension

The following table shows the Cortex-A78C features, which are updated from Cortex-A78.

Table A1-1 Cortex-A78C feature updates from Cortex-A78

Feature	Location
Added Pointer Authentication	• *B2.18 APDAKeyHi_EL1, Pointer Authentication Key A for Data* on page B2-167 • *B2.19 APDAKeyLo_EL1, Pointer Authentication Key A for Data* on page B2-168 • *B2.20 APDBKeyHi_EL1, Pointer Authentication Key B for Data* on page B2-169 • *B2.21 APDBKeyLo_EL1, Pointer Authentication Key B for Data* on page B2-170 • *B2.22 APGAKeyHi_EL1, Pointer Authentication Key A for Code* on page B2-171 • *B2.23 APGAKeyLo_EL1, Pointer Authentication Key A for Code* on page B2-172 • *B2.24 APIAKeyHi_EL1, Pointer Authentication Key A for Instruction* on page B2-173 • *B2.25 APIAKeyLo_EL1, Pointer Authentication Key A for Instruction* on page B2-174 • *B2.26 APIBKeyHi_EL1, Pointer Authentication Key B for Instruction* on page B2-175 • *B2.27 APIBKeyLo_EL1, Pointer Authentication Key B for Instruction* on page B2-176
Update to Hypervisor Configuration Register	*B2.74 HCR_EL2, Hypervisor Configuration Register, EL2* on page B2-256
Update to AArch64 Memory Model Feature Register 2	*B2.83 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1* on page B2-271
Update to AArch64 Instruction Set Attribute Registers	• *B2.79 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1* on page B2-263 • *B2.80 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1* on page B2-265
Update to Multiprocessor Affinity Register	*B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314
Updates to System Control Registers	• *B2.113 SCTLR_EL1, System Control Register, EL1* on page B2-320 • *B2.114 SCTLR_EL2, System Control Register, EL2* on page B2-323 • *B2.115 SCTLR_EL3, System Control Register, EL3* on page B2-325
Updates to Translation Control Registers	• *B2.116 TCR_EL1, Translation Control Register, EL1* on page B2-327 • *B2.117 TCR_EL2, Translation Control Register, EL2 Primes* on page B2-329 • *B2.118 TCR_EL3, Translation Control Register, EL3* on page B2-330

A1.3 Implementation options

All Cortex-A78C cores in the cluster must have the same configuration parameters, but each core can have a different L2 cache size.

The following table lists the implementation options for a core.

Table A1-2 Core implementation options

Feature	Range of options	Notes
L1 data cache size	• 32KB • 64KB	-
L1 instruction cache size	• 32KB • 64KB	-
L2 cache size	• 256KB • 512KB	-
L2 transaction queue size	• 48 entries • 56 entries • 62 entries	There are two identical L2 banks in the Cortex-A78C core that can be configured with 24, 28 or 31 L2 transaction queue entries per L2 bank.
L1 and L2 ECC or parity protection	Can be included or not included.	L1 and L2 error protection can only be enabled if L3 cache error protection is enabled.
Cryptographic Extension	Can be included or not included.	The Cryptographic Extension is a separately licensable product.
Core bus width	128-bit, 256-bit	This specifies the bus width between the core and the DSU CPU bridge. The legal core bus width and master bus width combinations are: • If the core bus width is 128 bits, the master bus interface can be any of the following options: — Single 128-bit wide ACE interface — Dual 128-bit wide ACE interfaces — Single 128-bit wide CHI interface — Single 256-bit wide CHI interface — Dual 256-bit wide CHI interface • If the core bus width is 256 bits, the master bus interface is a single or dual 256-bit wide CHI interface.
CoreSight *Embedded Logic Analyzer* (ELA)	Optional support	Support for integrating CoreSight ELA-500. CoreSight ELA-500 is a separately licensable product.
ELA RAM Address size	See the *Arm® CoreSight™ ELA-500 Embedded Logic Analyzer Technical Reference Manual* for the full supported range.	-

Note

Cache indices are determined such that the physical address and set number may not be directly correlated. A software flush using set and way operations that walk the entire space will work. Targeted operations that assume a relationship between the physical address and set number cannot be used. This is compliant with the Armv8-A architecture.

A1.4 Supported standards and specifications

The Cortex-A78C core implements the Armv8-A architecture and some architecture extensions. It also supports interconnect, interrupt, timer, debug, and trace architectures.

Table A1-3 Compliance with standards and specifications

Architecture specification or standard	Version	Notes
Arm architecture	Armv8-A	• AArch32 Execution state at Exception level EL0 only. AArch64 Execution state at all Exception levels (EL0-EL3). • A64, A32, and T32 instruction sets
Arm architecture extensions	• Armv8.1-A extensions • Armv8.2-A extensions • Cryptographic Extension • *Reliability, Availability, and Serviceability* (RAS) Extension • Armv8.3-A extensions • Armv8.4-A dot product instructions • Armv8.5-A extensions • Armv8.6-A Enhanced Pointer Authentication	• The Cortex-A78C core implements the `LDAPR` instructions introduced in the Armv8.3-A extensions. • The Cortex-A78C core implements the `SDOT` and `UDOT` instructions introduced in the Armv8.4-A extensions. • The Cortex-A78C core implements the PSTATE *Speculative Store Bypass Safe* (SSBS) bit introduced in the Armv8.5-A extension.
Generic Interrupt Controller	• GICv3 • GICv4	-
Generic Timer	Armv8-A	64-bit external system counter with timers within each core
Performance Monitoring Unit	PMUv3	-
Debug	Armv8-A	With support for the debug features added by the Armv8.2-A extensions
CoreSight	CoreSightv3	-
Embedded Trace Macrocell	ETMv4.2	Instruction trace only

See *Additional reading* on page 21 for a list of architectural references.

A1.5 Test features

The Cortex-A78C core provides test signals that enable the use of both *Automatic Test Pattern Generation* (ATPG) and *Memory Built-In Self Test* (MBIST) to test the core processing logic and memory arrays.

For more information, see *DFT integration guidelines* in the *Arm® Cortex-A78C Core Configuration and Integration Manual*.

A1.6 Design tasks

The Cortex-A78C core is delivered as a synthesizable *Register Transfer Level* (RTL) description in Verilog HDL. Before you can use the Cortex-A78C core, you must implement it, integrate it, and program it.

A different party can perform each of the following tasks. Each task can include implementation and integration choices that affect the behavior and features of the core.

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

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

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

The operation of the final device depends on the following:

Build configuration
The implementer chooses the options that affect how the RTL source files are pre-processed. 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 core by tying inputs to specific values. These configuration settings affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

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

A1.7 Product revisions

This section indicates the first release and, in subsequent releases, describes the differences in functionality between product revisions.

r0p1 First release

Chapter A2
Technical overview

This chapter describes the structure of the Cortex-A78C core.

It contains the following sections:

A2.1 Components

In a standalone configuration, there can be up to four Cortex-A78C cores and a *DynamIQ Shared Unit* (DSU) that connects the cores to an external memory system.

For more information about the DSU components, see *Components* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

The main components of the Cortex-A78C core are:

- Instruction fetch
- Instruction decode
- Register rename
- Instruction issue
- Execution pipelines
- L1 data memory system
- L2 memory system

The following figure is an overview of the Cortex-A78C core.

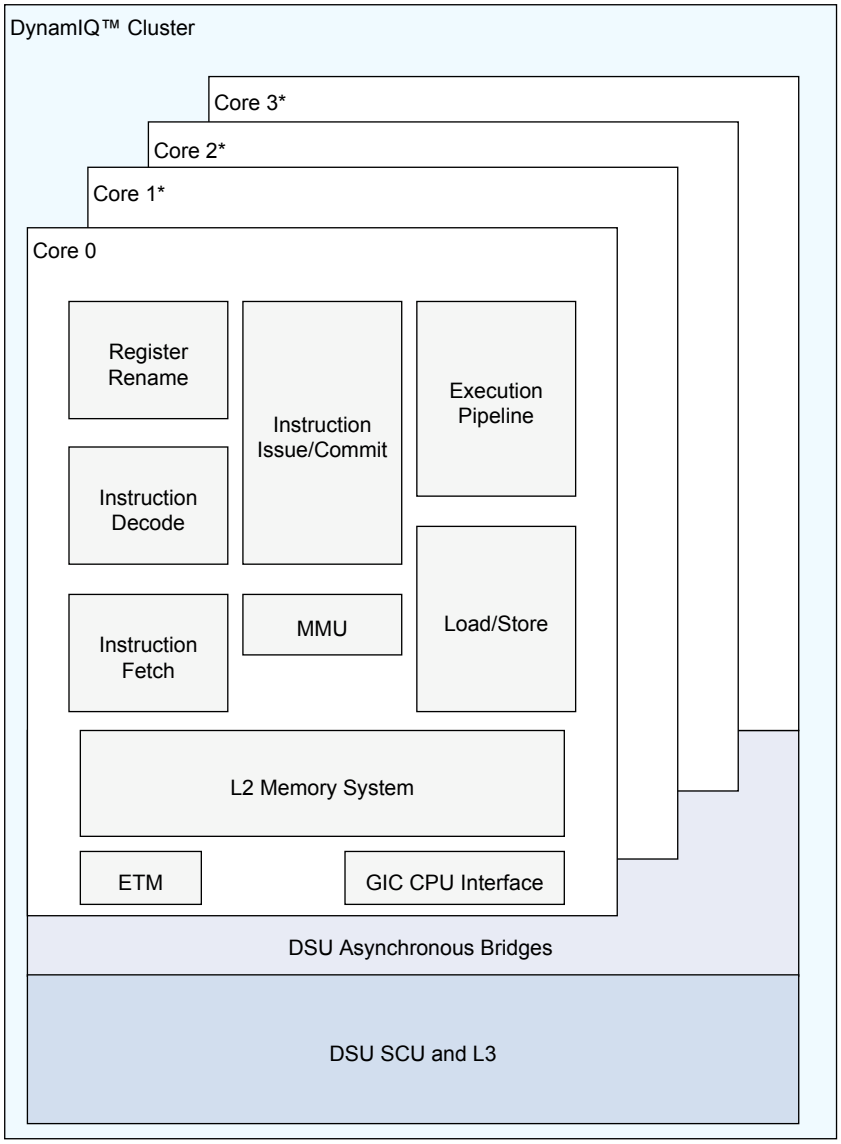

Figure A2-1 Cortex-A78C core overview

Note

There are multiple asynchronous bridges between the Cortex-A78C core and the DSU. Only the coherent interface between the Cortex-A78C core and the DSU can be configured to run synchronously, however it does not affect the other interfaces such as debug, trace, and *Generic Interrupt Controller* (GIC) which are always asynchronous. For more information on how to set the coherent interface to run either synchronously or asynchronously, see *Configuration Guidelines* in the *Arm® DynamIQ™ Shared Unit MP135 Configuration and Sign-off Guide*.

A2.1.1 Instruction fetch

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

The instruction fetch unit includes:

- A 4-way, set associative L1 instruction cache with 64-byte cache lines and optional parity protection. The L1 instruction cache is configurable with sizes of 32KB or 64KB.
- A fully associative L1 instruction TLB with native support for 4KB, 16KB, 64KB, and 2MB page sizes.
- A 1.5K entry, 4-way skewed associative L0 Macro-OP (MOP) cache with optional parity, which contains decoded and optimized instructions for higher performance.
- A dynamic branch predictor.

A2.1.2 Instruction decode

The instruction decode unit supports the A32, T32, and A64 instruction sets. It also supports Advanced SIMD and floating-point instructions in each instruction set.

A2.1.3 Register rename

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

A2.1.4 Instruction issue

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

A2.1.5 Execution pipeline

The execution pipeline includes:

- Integer execute unit that performs arithmetic and logical data processing operations.
- Vector execute unit that performs Advanced SIMD and floating-point operations. Optionally, it can execute the cryptographic instructions.

A2.1.6 L1 data memory system

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

The load/store unit includes:

- A 4-way, set associative L1 data cache with 64-byte cache lines and optional ECC protection per 32 bits. The L1 data cache is configurable with sizes of 32KB or 64KB.
- A fully associative L1 data TLB with native support for 4KB, 16KB, 64KB page sizes and 2MB, 512MB block sizes.

A2.1.7 L2 memory system

The L2 memory system services L1 instruction and data cache misses in the Cortex-A78C core.

The L2 memory system includes:

- An 8-way set associative L2 cache with data *Error Correcting Code* (ECC) protection per 64 bits. The L2 cache is configurable with sizes of 256KB, or 512KB.
- An interface with the DSU configurable at implementation time for synchronous or asynchronous operation.

For direct accesses to the contents of the L2 RAMs where the index and way is specified for set and way operations, the mechanism to compute the cache indices from the physical address is shown in the following table.

L2 cache size	Physical address
256KB	addr[22:15] XOR addr[14:7]
512KB	addr[24:16] XOR addr[15:7]

Related references

Chapter A5 Memory Management Unit on page A5-65
Chapter A6 L1 memory system on page A6-75
Chapter A7 L2 memory system on page A7-103
Chapter A9 Generic Interrupt Controller CPU interface on page A9-117
Chapter C1 Debug on page C1-419
Chapter C2 Performance Monitoring Unit on page C2-425
Chapter C4 Embedded Trace Macrocell on page C4-445

A2.2 Interfaces

The Cortex-A78C core has several interfaces to connect it to a *System on Chip* (SoC). The *DynamIQ Shared Unit* (DSU) manages all interfaces.

For information on the interfaces, see *Technical overview* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

A2.3 About system control

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

The main functions of the System registers are:

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

The system registers are accessible in the AArch64 EL0-EL3 and AArch32 EL0 Execution state. Some of the System registers are accessible through the external debug interface.

A2.4 About the Generic Timer

The Generic Timer can schedule events and trigger interrupts that are based on an incrementing counter value. It generates timer events as active-LOW interrupt outputs and event streams.

The Cortex-A78C core provides a set of timer registers. The timers are:

- An EL1 Non-secure physical timer
- An EL2 Hypervisor physical timer
- An EL3 Secure physical timer
- A virtual timer
- A Hypervisor virtual timer

The Cortex-A78C core does not include the system counter. This resides in the *System on Chip* (SoC). The system counter value is distributed to the core over a 64-bit bus.

For more information on the Generic Timer, see the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual* and the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Chapter A3
Clocks, resets, and input synchronization

This chapter describes the clocks, resets, and input synchronization of the Cortex-A78C core.

It contains the following sections:

A3.1 About clocks, resets, and input synchronization

The Cortex-A78C core supports hierarchical clock gating.

The Cortex-A78C core contains several interfaces that connect to other components in the system. These interfaces can be in the same clock domain or in other clock domains.

For information about clocks, resets, and input synchronization, see *Functional description* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

A3.2 Asynchronous interface

Your implementation can include an optional asynchronous interface between the core and the *DynamIQ Shared Unit* (DSU) top level.

See *Implementation options* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual* for more information.

Chapter A4
Power management

This chapter describes the power domains and the power modes in the Cortex-A78C core.

It contains the following sections:

A4.1 About power management

The Cortex-A78C core provides mechanisms to control both dynamic and static power dissipation.

Dynamic power management includes the following features:

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

Static power management includes the following features:

- Dynamic retention
- Powerdown

Related references

A4.3 Power domains on page A4-52
A4.5 Power control on page A4-56
A4.8 Power domain states for power modes on page A4-61
A4.9 Core powerup and powerdown sequences on page A4-62

A4.2 Voltage domains

The Cortex-A78C core supports a VCPU voltage domain and a VSYS voltage domain.

The following figure shows the VCPU and VSYS voltage domains in each Cortex-A78C core and in the *DynamIQ Shared Unit* (DSU). The example shows a configuration with four Cortex-A78C cores.

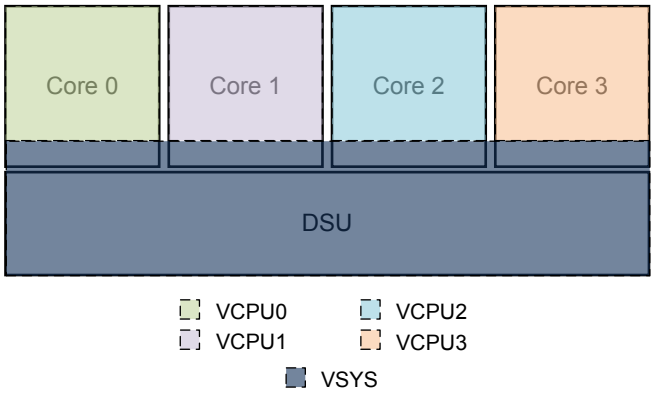

Figure A4-1 Cortex-A78C voltage domains

Asynchronous bridge logic exists between the voltage domains. The Cortex-A78C core processing logic and core clock domain of the asynchronous bridge are in the VCPU voltage domain. The DSU clock domain of the asynchronous bridge is in the VSYS voltage domain.

Note

You can tie VCPU and VSYS to the same supply if the core is not required to support *Dynamic Voltage and Frequency Scaling* (DVFS). The core can still be powered down independently if it is in its own power domain with proper isolation.

A4.3 Power domains

The Cortex-A78C core contains a Core power domain (PDCPU) and a core top-level SYS power domain (PDSYS) where all the Cortex-A78C core I/O signals go through.

PDCPU power domain

The PDCPU power domain contains all core processing logic excluding the cluster clock domain side of the bridge.

PDSYS power domain

The PDSYS power domain contains the cluster clock domain side of the bridge.

Note

There are additional system power domains in the *DynamIQ Shared Unit* (DSU). See the *Power management* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual* for information.

The following figure shows an example of how the voltage and power domains are organized.

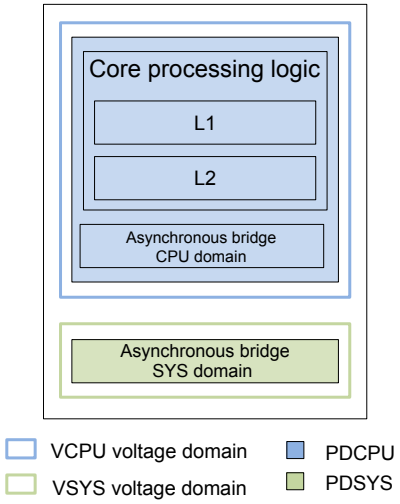

Figure A4-2 Cortex-A78C core power domain diagram

The following table describes the power domains that the Cortex-A78C core supports.

Table A4-1 Power domain description

Power domain	Description
PDCPU<n>	The domain includes the Advanced SIMD and floating-point block, the L1 and L2 TLBs, L1 and L2 cache RAMs, and Debug registers that are associated with the Cortex-A78C core. <n> is the core number in the range 0-3. The number represents core 0, core 1, core 2, and core 3. If a core is not present, then the corresponding power domain is not present.
PDSYS	The domain is the interface between Cortex-A78C and the DSU. It contains the cluster clock domain logic of the CPU bridge. The CPU bridge contains all asynchronous bridges for crossing clock domains, and is split with one half of each bridge in the core clock domain and the other half in the relevant cluster domain. All core I/O signals go through the CPU bridge and the SYS power domain.

Clamping cells between power domains are inferred through power intent files rather than instantiated in the RTL.

The following figure shows the power domains in the DSU cluster, where everything in the same color is part of the same power domain. The number of Cortex-A78C cores can vary, and the number of domains increases based on the number of Cortex-A78C cores present. This example only shows four Cortex-A78C cores and the power domains that are associated with them. Other power domains are required for a DSU cluster and are not shown in this example.

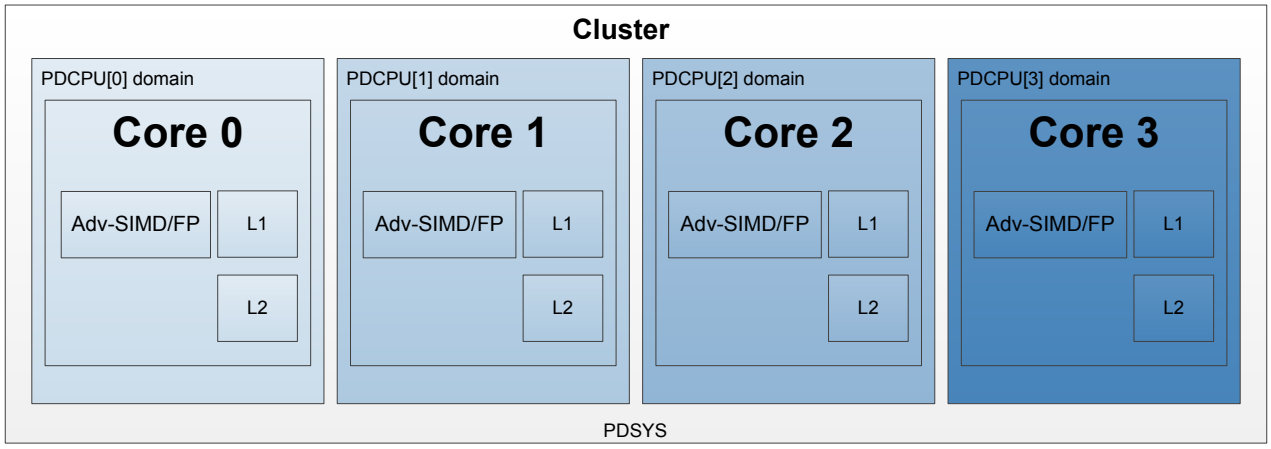

Figure A4-3 Cortex-A78C power domains

A4.4 Architectural clock gating modes

When the Cortex-A78C core is in Standby mode, it is architecturally clock gated at the top of the clock tree.

Wait For Interrupt (WFI) and *Wait For Event* (WFE) are features of Armv8-A architecture that put the core in a low-power Standby mode by architecturally disabling the clock at the top of the clock tree. The core is fully powered and retains all the state in Standby mode.

A4.4.1 Core Wait for Interrupt

Wait For Interrupt (WFI) uses a locking mechanism, based on events, to put the core in a low-power state by disabling most of the clocks in the core, while keeping the core powered up.

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

While the core is in WFI low-power state, the clocks in the core are temporarily enabled without causing the core to exit WFI low-power state when any of the following events are detected:

- A cluster snoop request that must be serviced by the core data caches
- A cache or *Translation Lookaside Buffer* (TLB) maintenance operation that must be serviced by the core L1 instruction cache, data cache, TLB, or L2 cache
- An *Advanced Peripheral Bus* (APB) access to the debug or trace registers residing in the core power domain
- A GIC CPU access through the AXI4-Stream channel

Exit from WFI low-power state occurs when the core detects one of the following:

- A reset
- One of the WFI wake up events

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

A4.4.2 Core Wait for Event

Wait For Event (WFE) uses a locking mechanism, based on events, to put the core in a low-power state by disabling most of the clocks in the core, while keeping the core powered up.

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

If the event register is set, execution of WFE does not cause entry into standby state, but clears the event register.

While the core is in WFE low-power state, the clocks in the core are temporarily enabled without causing the core to exit WFE low-power state when any of the following events are detected:

- A cluster snoop request that must be serviced by the core data caches
- A cache or *Translation Lookaside Buffer* (TLB) maintenance operation that must be serviced by the core L1 instruction cache, data cache, TLB, or L2 cache
- An *Advanced Peripheral Bus* (APB) access to the debug or trace registers residing in the core power domain
- A GIC CPU access through the AXI4-Stream channel

Exit from WFE low-power state occurs when one of the following occurs:

- The core detects one of the WFE wake up events.
- The **EVENTI** input signal is asserted.
- The core detects a reset.

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

A4.5 Power control

All power mode transitions are performed at the request of the power controller, using a P-Channel interface to communicate with the Cortex-A78C core.

There is one P-Channel per core, plus one P-Channel for the cluster. The Cortex-A78C core provides the current requirements on the **PACTIVE** signals, so that the power controller can make decisions and request any change with **PREQ** and **PSTATE**. The Cortex-A78C core then performs any actions necessary to reach the requested power mode, such as gating clocks, flushing caches, or disabling coherency, before accepting the request.

If the request is not valid, either because of an incorrect transition or because the status has changed so that state is no longer appropriate, then the request is denied. The power mode of each core can be independent of other cores in the cluster, however the cluster power mode is linked to the mode of the cores.

A4.6 Core power modes

The following figure shows the supported modes for each core domain P-Channel, and the legal transitions between them.

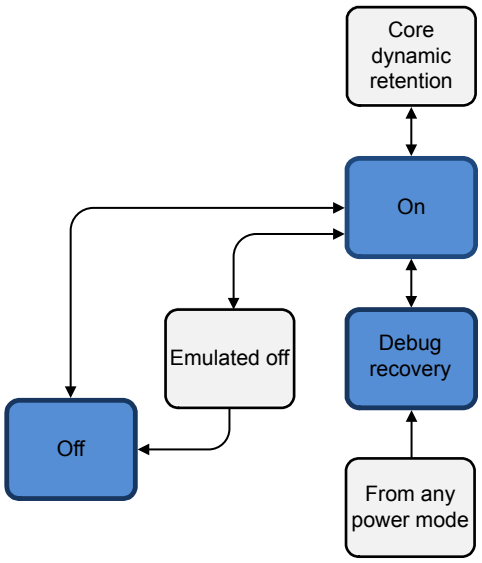

Figure A4-4 Cortex-A78C core power domain mode transitions

The blue modes indicate the modes the channel can be initialized into.

A4.6.1 On mode

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

The core can be initialized into the On mode. If the core does not use P-Channel, you can tie the core in the On mode by tying **PREQ** LOW.

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.

When the core domain P-Channel is initialized into the On mode, either as a shortcut for entering that mode or as a tie-off for an unused P-Channel, it is an assumed transition from the Off mode. This includes an invalidation of any cache RAM within the core domain.

A4.6.2 Off mode

The Cortex-A78C core supports a full Shutdown mode where power can be removed completely and no state is retained.

The shutdown can be for either the whole cluster or just for an individual core, which allows other cores in the cluster to continue operating.

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

A Cold reset can reset the core in this mode.

The core P-Channel can be initialized into this mode.

An attempted debug access when the core domain is off returns an error response on the internal debug interface, indicating that the core is not available.

A4.6.3 Emulated off mode

In this mode, all core domain logic and RAMs are kept on. However, core Warm reset can be asserted externally to emulate a power off scenario while keeping core debug state and allowing debug access.

All debug registers must retain their mode and be accessible from the external debug interface. All other functional interfaces behave as if the core were Off.

A4.6.4 Core dynamic retention mode

In this mode, all core processing logic and RAMs are in retention and the core domain is inoperable. The core can be entered into this power mode when it is in *Wait For Interrupt* (WFI) or *Wait For Event* (WFE) mode.

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

- Use the CPUPWRCTLR.WFI_RET_CTRL register bits to program timeout values for entry into core dynamic retention mode from WFI mode.
- Use the CPUPWRCTLR.WFE_RET_CTRL register bits to program timeout values for entry into core dynamic retention mode from WFE mode.

When in dynamic retention and the core is synchronous to the cluster, the clock to the core is automatically gated outside of the domain. However, if the core is running asynchronous to the cluster, the system integrator must gate the clock externally during core dynamic retention. For more information, see the *Arm® DynamIQ™ Shared Unit MP135 Configuration and Sign-off Guide*.

The outputs of the domain must be isolated to prevent buffers without power from propagating UNKNOWN values to any operational parts of the system.

When the core is in dynamic retention there is support for snoop, GIC, and debug access, so the core appears as if it were in WFI or WFE mode. When such an incoming access occurs, it stalls and the On **PACTIVE** bit is set HIGH. The incoming access proceeds when the domain is returned to On using the P-Channel.

When the incoming access completes, and if the core has not exited WFI or WFE mode, then the On **PACTIVE** bit is set LOW after the programmed retention timeout. The power controller can then request to reenter the core dynamic retention mode.

A4.6.5 Debug recovery mode

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

It allows contents of the core L1 instruction, L1 data and L2 caches that were present before the reset to be observable after the reset. The contents of the caches are retained and are not altered on the transition back to the On mode.

By default, the core invalidates its caches when transitioning from Off to On mode. If the P-Channel is initialized to debug recovery, and the core is cycled through Cold or Warm reset along with system resets, then the cache invalidation is disabled. The cache contents are preserved when the core is transitioned to the On mode.

Debug recovery also supports preserving *Reliability, Availability, and Serviceability* (RAS) state, in addition to the cache contents. In this case, a transition to debug recovery is made from any of the current states. Once in Debug recovery mode, a cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to the On mode.

——— Caution ———

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

- This mode can occur at any time with no guarantee of the state of the core. A 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 at the time of the reset, then these might complete after the reset when the core 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, or it might not get a response and cause a system deadlock.

A4.7 Encoding for power modes

The following table shows the encodings for the supported modes for each core domain P-Channel.

Table A4-2 Core power modes COREPSTATE encoding

Power mode	Short name	PACTIVE bit number	PSTATE value	Power mode description
Core debug recovery mode	DEBUG_RECOV	-	`0b001010`	Logic is off (or in reset), RAM state is retained and not invalidated when transitioning to On mode.
On mode	ON	8	`0b001000`	All powerup
Core dynamic retention mode	FULL_RET	5	`0b000101`	Logic and RAM state are inoperable but retained.
Emulated off mode	OFF_EMU	1	`0b000001`	On with Warm reset asserted, Debug state is retained and accessible.
Off mode	OFF	0 (implicit)[a]	`0b000000`	All powerdown

[a] It is tied off to 0 and should be inferred when all other PACTIVE bits are LOW. For more information, see the *AMBA® Low Power Interface Specification Arm® Q-Channel and P-Channel Interfaces*.

A4.8 Power domain states for power modes

The power domains can be controlled independently to give different combinations when powered-up and powered-down.

However, only some powered-up and powered-down domain combinations are valid and supported. The following information shows the supported power domain states for the Cortex-A78C core.

The PDCPU power domain supports the power states described in the following table.

Table A4-3 Power state description

Power state	Description
Off	Core off. Power to the block is gated.
Ret	Core retention. Logic and RAM retention power only.
On	Core on. Block is active.

Caution

States that are not shown in the following tables are unsupported and must not occur.

The following table describes the power modes, and the corresponding power domain states for individual cores. The power mode of each core is independent of all other cores in the cluster.

Table A4-4 Supported core power domain states

Power mode	Power domain state	Description
Debug recovery	On	Core on
On	On	Core on
Core dynamic retention	Ret	Core in retention
Off (emulated)	On	Core on
Off	Off	Core off

Deviating from the legal power modes can lead to UNPREDICTABLE results. You must comply with the dynamic power management and powerup and powerdown sequences described in the following sections.

A4.9 Core powerup and powerdown sequences

The following approach allows taking the Cortex-A78C cores in the cluster in and out of coherence.

Core powerdown

To take a core out of coherence ready for core powerdown:

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

All L1 and L2 cache disabling, L1 and L2 cache flushing, and communication with the L3 memory system is performed in hardware after the `WFI` is executed, under the direction of the power controller.

Note

Executing any `WFI` instruction when the CPUPWRCTLR.CORE_PWRDN_EN bit is set automatically masks out all interrupts and wake up events in the core. If executed when the CPUPWRCTLR.CORE_PWRDN_EN bit is set the WFI never wakes up and the core needs to be reset to restart.

For information about cluster powerdown, see *Power management* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

Core powerup

To bring a core into coherence after reset, no software steps are required.

Related references

B2.54 CPUPWRCTLR_EL1, Power Control Register, EL1 on page B2-231

A4.10 Debug over powerdown

The Cortex-A78C core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This enables debug to continue through powerdown scenarios, rather than having to re-establish a connection each time the core is powered up.

The debug over powerdown logic is part of the DebugBlock, which is external to the cluster and can be implemented in a separate power domain. If the DebugBlock is in the same power domain as the core, then debug over powerdown is not supported.

For more information on the DebugBlock, see *Debug* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

Chapter A5
Memory Management Unit

This chapter describes the *Memory Management Unit* (MMU) of the Cortex-A78C core.

It contains the following sections:

A5.1 About the 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.

A5.1.1 Main functions

The three main functions of the *Memory Management Unit* (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 called translation tables. Translation table entries can be cached into a *Translation Lookaside Buffer* (TLB).

The following table describes the components included in the MMU.

Table A5-1 TLBs and TLB caches in the MMU

Component	Description
Instruction L1 TLB	32 entries, fully associative
Data L1 TLB	32 entries, fully associative
L2 TLB cache	1024 entries, 4-way set associative
Translation table prefetcher	Detects access to contiguous translation tables and prefetches the next one. This prefetcher can be disabled in the ECTLR register.

The TLB entries contain either one or both of a global indicator and an *Address Space Identifier* (ASID) to permit context switches without requiring the TLB to be invalidated.

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

A5.1.2 AArch64 behavior

The Cortex-A78C core is an Armv8 compliant core that supports execution in AArch64 state.

The following table shows the AArch64 behavior.

Table A5-2 AArch64 behavior

Feature	Description
Address translation system	The Armv8 address translation system resembles an extension to the Long descriptor format address translation system to support the expanded virtual and physical address space.
Translation granule	4KB, 16KB, or 64KB for Armv8 AArch64 *Virtual Memory System Architecture* (VMSAv8-64). Using a larger granule size can reduce the maximum required number of levels of address lookup.
ASID size	8-bit or 16-bit depending on the value of TCR_ELx.AS.

Table A5-2 AArch64 behavior (continued)

Feature	Description
VMID size	8-bit or 16-bit depending on the value of VTCR_EL2.VS.
PA size	Maximum 40 bits. Any configuration of TCR_ELx.IPS over 40 bits is considered as 40 bits. You can enable or disable each stage of the address translation independently.

The Cortex-A78C core also supports the *Virtualization Host Extension* (VHE) including ASID space for EL2. When VHE is implemented and enabled, EL2 has the same behavior as EL1.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on concatenated translation tables and for address translation formats.

A5.2 TLB organization

The *Translation Lookaside Buffer* (TLB) is a cache of recently executed page translations within the *Memory Management Unit* (MMU). The Cortex-A78C core implements a two-level TLB structure. The TLB stores all page sizes and is responsible for breaking these down in to smaller pages when required for the data or instruction L1 TLB.

A5.2.1 Instruction L1 TLB

The instruction L1 TLB is implemented as a 32-entry fully associative structure. This TLB caches entries at the 4KB, 16KB, 64KB, and 2MB granularity of VA to PA mapping only.

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

A5.2.2 Data L1 TLB

The data L1 TLB is a 32-entry fully associative TLB that is used by load and store operations. The cache entries have 4KB, 16KB, 64KB, 2MB, and 512MB granularity of VA to PA mappings only.

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

A5.2.3 L2 TLB

The L2 TLB structure is shared by instruction and data. It handles misses from the instruction and data L1 TLBs.

The following table describes the L2 TLB characteristics.

Table A5-3 Characteristic of the L2 TLB

Characteristic	Note
4-way, set associative, 1024-entry cache	Stores: • VA to PA mappings for 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, and 1GB block sizes. • *Intermediate physical address* (IPA) to PA mappings for 2MB and 1GB (in a 4KB translation granule), 32MB (in a 16K translation granule), and 512MB (in a 64K granule) block sizes. Only Non-secure EL1 and EL0 stage 2 translations are cached. • Intermediate PAs obtained during a translation table walk.

Access to the L2 TLB usually takes three cycles. If a different page or block size mapping is used, then this access can take longer.

The L2 TLB supports four translation table walks in parallel (four TLB misses), and can service two TLB lookups while the translation table walks are in progress. If there are six successive misses, the L2 TLB will stall.

Note

Caches in the core are invalidated automatically at reset deassertion unless the core power mode is initialized to Debug recovery mode. See *Power management* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual* for more information.

A5.3 TLB match process

The Armv8-A architecture provides support for multiple maps from the *Virtual Addresses* (VAs) space that are translated differently.

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

Each TLB entry contains:

- VA
- *Physical Addresses* (PAs)
- Set of memory properties that include type and access permissions

Each entry is either associated with a particular *Address Space Identifier* (ASID) or 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 entry is associated with a particular translation regime:

- EL3 in Secure state in AArch64 state only
- EL2 or EL0 in Non-secure state
- EL1 or EL0 in Secure state
- EL1 or EL0 in Non-secure state

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

- VA bits[48:N], where N is $\log_2$ of the block size for that translation that is stored in the TLB entry, matches the requested address.
- Entry translation regime matches the current translation regime.
- The ASID matches the current ASID held in the CONTEXTIDR, TTBR0, or TTBR1 register, or the entry is marked global.
- The VMID matches the current VMID held in the VTTBR_EL2 register.
- The ASID and VMID matches are IGNORED when ASID and VMID are not relevant.
  ASID is relevant when the translation regime is:
  — EL2 in Non-secure state with HCR_EL2.E2H and HCR_EL2.TGE set to 1
  — EL1 or EL0 in Secure state
  — EL1 or EL0 in Non-secure state

  VMID is relevant for EL1 or EL0 in Non-secure state.

A5.4 Translation table walks

When an access is requested at an address, the *Memory Management Unit* (MMU) searches for the requested *Virtual Address* (VA) in the *Translation Lookaside Buffers* (TLB). 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-A78C core generates a memory access, the following process occurs:

1. The MMU performs a lookup for the requested VA, current *Address Space Identifier* (ASID), current *Virtual Machine Identifier* (VMID), and current translation regime in the relevant instruction or data L1 TLB.
2. If there is a miss in the relevant L1 TLB, the MMU performs a lookup 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 a 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 core returns a Translation Fault. If the TLB finds a matching entry, it uses the information in the entry as follows.

The access permission bits 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 Armv8, for Armv8-A architecture profile* 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

This section contains the following subsection:

- *A5.4.1 AArch64 behavior* on page A5-70.

A5.4.1 AArch64 behavior

When executing in AArch64 state at a particular Exception level, you can configure the hardware translation table walk to use either the 4KB, 16KB, or 64KB translation granule.

Program the Translation Granule bit, TG0, in the appropriate translation control register:

- TCR_EL1
- TCR_EL2
- TCR_EL3
- VTCR_EL2

For TCR_EL1, you can program the Translation Granule bits TG0 and TG1 to configure the translation granule respectively for TTBR0_EL1 and TTBR1_EL1, or TCR_EL2 when *Virtualization Host Extension* (VHE) is enabled.

A5.5 MMU memory accesses

During a translation table walk, the *Memory Management Unit* (MMU) generates accesses. This section describes the specific behaviors of the core for MMU memory accesses.

A5.5.1 Configuring MMU accesses

By programming the IRGN and ORGN bit fields of the appropriate TCR_EL*x* registers, you can configure the *Memory Management Unit* (MMU) to perform translation table walks in cacheable or Non-cacheable regions.

If the encoding of both the ORGN and IRGN bit fields is Write-Back, the data cache lookup is performed and data is read from the data cache. External memory is accessed, if the ORGN and IRGN bit fields contain different attributes, or if the encoding of the ORGN and IRGN bit fields is Write-Through or Non-cacheable.

A5.5.2 Descriptor hardware update

The core supports hardware update in AArch64 state using hardware management of the access flag and hardware management of dirty state.

These features are enabled in registers TCR_ELx and VTCR_EL2.

Hardware management of the Access flag is enabled by the following configuration fields:

- TCR_ELx.HA for stage 1 translations
- VTCR_EL2.HA for stage 2 translations

Hardware management of dirty state is enabled by the following configuration fields:

- TCR_ELx.HD for stage 1 translations
- VTCR_EL2.HD for stage 2 translations

Note

Hardware management of dirty state can only be enabled if hardware management of the Access flag is enabled.

To support the hardware management of dirty state, the *Dirty Bit Modifier* (DBM) field is added to the translation table descriptors as part of Armv8.1 architecture.

The core supports hardware update only in outer Write-Back and inner Write-Back memory regions.

If software requests a hardware update in a memory region that is not inner Write-Back or not outer Write-Back, then the core returns an abort with the following encoding:

- ESR.ELx.DFSC = `0b110001` for Data Aborts in AArch64
- ESR.ELx.IFSC = `0b110001` for Instruction Aborts in AArch64

A5.6 Specific behaviors on aborts and memory attributes

This section describes specific behaviors caused by aborts and also describes memory attributes.

MMU responses

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

- A L1 TLB hit
- A L2 TLB hit
- A translation table walk

The response from the MMU contains the following information:

- The PA corresponding to the translation
- A set of permissions
- Secure or Non-secure
- All the information required to report aborts

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

A5.6.1 External aborts

External aborts are defined as those that occur in the memory system rather than those that the *Memory Management Unit* (MMU) detects. Normally, external memory aborts are rare. External aborts are caused by errors flagged to the external interface.

External aborts are reported synchronously when they occur during translation table walks, data access due to loads to Normal memory, loads with acquire semantics to Device memory and LD/CAS atomics. The address captured in the fault address register is that of the address that generated the synchronous external abort. External aborts are reported asynchronously when they occur for loads to Device memory, stores to any memory type, for cache maintenance, TLB invalidate, and instruction cache invalidate operations.

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

A5.6.2 Mis-programming contiguous hints

In the case of a mis-programming contiguous hint, when there is a descriptor that contains a set CH bit, all contiguous VAs contained in this block should be included in the input VA address space that is defined for stage 1 by TxSZ for TTBx or for stage 2 by {SL0, T0SZ}.

The Cortex-A78C core treats such a block as not causing a translation fault.

A5.6.3 Memory attributes

The memory region attributes specified in the TLB entry, or in the descriptor in case of translation table walk, determine if the access is:

- Normal Memory or Device type
- One of the four different device memory types that are defined for Armv8:

Device-nGnRnE	Device non-Gathering, non-Reordering, No Early Write Acknowledgement
Device-nGnRE	Device non-Gathering, non-Reordering, Early Write Acknowledgement
Device-nGRE	Device non-Gathering, Reordering, Early Write Acknowledgement
Device-GRE	Device Gathering, Reordering, Early Write Acknowledgment

In the Cortex-A78C core, a page is cacheable only if the inner memory attribute and outer memory attribute are write-back. In all other cases, all pages are downgraded to Non-cacheable Normal memory.

When the MMU is disabled at stage 1 and stage 2, and SCTLR.I is set to 1, instruction prefetches are cached in the instruction cache but not in the unified cache. In all other cases, normal behavior on memory attribute applies.

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

A5.7 Page-based hardware attributes

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

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

For information on how to set and enable the PBHA bits in the translation tables, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. When disabled, the PBHA value that is propagated on the bus is 0.

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

PBHA combination between stage 1 and stage 2 on memory accesses

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

When stage 1 and stage 2 are enabled:

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

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

Mismatched aliases

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

Chapter A6
L1 memory system

This chapter describes the L1 instruction cache and data cache that make up the L1 memory system.

It contains the following sections:

A6.1 About the L1 memory system

The Cortex-A78C L1 memory system is designed to enhance core performance and save power.

The L1 memory system consists of separate instruction and data caches. Both are independently configurable to be either 32KB or 64KB.

A6.1.1 L1 instruction-side memory system

The L1 instruction memory system has the following key features:

- *Virtually Indexed, Physically Tagged* (VIPT) 4-way set-associative L1 instruction cache, which behaves as a *Physically Indexed, Physically Tagged* (PIPT) cache
- Fixed cache line length of 64 bytes
- Pseudo-LRU cache replacement policy
- 256-bit read interface from the L2 memory system

The Cortex-A78C core also has a *Virtually Indexed, Virtually Tagged* (VIVT) 4-way skewed-associative, *Macro-OP* (MOP) cache, which behaves as a PIPT cache.

A6.1.2 L1 data-side memory system

The L1 data memory system has the following features:

- *Virtually Indexed, Physically Tagged* (VIPT), which behaves as a *Physically Indexed, Physically Tagged* (PIPT) 4-way set-associative L1 data cache
- Fixed cache line length of 64 bytes
- Pseudo-LRU cache replacement policy
- 512-bit write interface from the L2 memory system
- 512-bit read interface from the L2 memory system
- Three 128-bit read paths from the data L1 memory system to the datapath
- 256-bit write path from the datapath to the L1 memory system

A6.2 Cache behavior

The IMPLEMENTATION SPECIFIC features of the instruction and data caches include:

- At reset the instruction and data caches are disabled and both caches are automatically invalidated.

Note

Caches in the core are invalidated automatically at reset deassertion unless the core power mode is initialized to Debug recovery mode. For more information, see the *Power management* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

- You can enable or disable each cache independently.
- Cache lockdown is not supported.
- On a cache miss, data for the cache linefill is requested in critical word-first order.

A6.2.1 Instruction cache disabled behavior

If the instruction cache is disabled, fetches cannot access any of the instruction cache arrays. An exception is the instruction cache maintenance operations. If the instruction cache is disabled, the instruction cache maintenance operations can still execute normally.

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

A6.2.2 Instruction cache speculative memory accesses

Instruction fetches are speculative. Execution is not guaranteed, because there can be several unresolved branches in the pipeline.

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

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

If the instruction cache is enabled, and if the instruction fetches miss in the L1 instruction cache, they can still look up in the L1 data caches. However, a new line is not allocated in the data cache unless the data cache is enabled.

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

A6.2.3 Data cache disabled behavior

If the data cache is disabled, load and store instructions do not access any of the L1 data, L2 cache, and if present, the *DynamicIQ Shared Unit* (DSU) L3 cache arrays.

Unless the data cache is enabled, a new line is not allocated in the L2 or L3 caches due to an instruction fetch.

Data cache maintenance operations are an exception. If the data cache is disabled, the data cache maintenance operations execute normally.

If the data cache is disabled, all loads 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 account for this possibility.

The L2 and L1 data caches cannot be disabled independently.

A6.2.4 Data cache maintenance considerations

DC IVAC operations in AArch64 state are treated as DC CIVAC except for permission checking and watchpoint matching.

DC IMVAC operations in AArch32, and DC IVAC instructions in AArch64, perform an invalidate of the target address. If the data is dirty within the cluster, a clean is performed before the invalidate.

DC ISW operations in AArch32, and DC ISW instructions in AArch64, perform both a clean and invalidate of the target set/way. The values of HCR.SWIO and HCR_EL2.SWIO have no effect.

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

A6.2.5 Data cache coherency

To maintain data coherency between multiple cores, the Cortex-A78C core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

A6.2.6 Write streaming mode

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

However, there are some situations where allocating on writes is not required. For example, when executing the C standard library `memset`() function to clear a large block of memory to a known value. 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 L1 memory system includes logic to detect when the core has stores pending to a full cache line when it is waiting for a linefill to complete, or when it detects a `DCZVA` (full cache line write to zero). If this situation is detected, then it switches into write streaming mode.

When in write streaming mode, loads behave as normal, and can still cause linefills, and writes still lookup in the cache, but if they miss then they write out to L2 (or possibly L3, system cache, or DRAM) rather than starting a linefill.

The L1 memory system continues in write streaming mode until it can no longer create a full cacheline of stores (for example because of a lack of resource in the L1 memory system) or has detected a high proportion of stores hitting in the cache.

Note

The L1 memory system is monitoring transaction traffic through L1 and, depending on different thresholds, can set a stream to go out to L2 cache, L3 cache, and system cache and DRAM.

The following register controls the different thresholds:

AArch64 state

CPUECTLR_EL1 configure the L2, L3, and system cache write streaming mode threshold. See *B2.44 CPUECTLR_EL1, CPU Extended Control Register, EL1* on page B2-205.

A6.3 L1 instruction memory system

The L1 instruction side memory system provides an instruction stream to the decoder.

It uses dynamic branch prediction and instruction caching to increase overall performance and to reduce power consumption.

A6.3.1 Program flow prediction

The Cortex-A78C core contains program flow prediction hardware, also known as branch prediction.

Branch prediction increases overall performance and reduces power consumption. With program flow 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. For conditional branches, the hardware predicts if the branch is to be taken. It also predicts the address that the branch goes to, known as the branch target address. For unconditional branches, only the target is predicted.

The hardware contains the following functionality:
- A *Branch Target Buffer* (BTB) holding the branch target address of previously taken branches
- A branch direction predictor using previous branch history
- The return stack, a stack of nested subroutine return addresses
- A static branch predictor
- An indirect branch predictor

Predicted and non-predicted instructions

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

Exception return instructions are not predicted.

T32 state conditional branches

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

Return stack

The return stack stores the address and instruction set state.

This address is equal to the link register value stored in R14 in AArch32 state or X30 in AArch64 state.

The following instructions cause a return stack push if predicted:
- `BL r14`
- `BLX (immediate)` in AArch32 state
- `BLX (register)` in AArch32 state
- `BLR` in AArch64 state
- `MOV pc,r14`

In AArch32 state, the following instructions cause a return stack pop if predicted:
- `BX`
- `LDR pc, [r13], #imm`
- `LDM r13, {…pc}`
- `LDM r13, {…pc}`

In AArch64 state, the RET instruction causes a return stack pop.

As exception return instructions can change core privilege mode and Security state, they are not predicted. These include ERET instruction.

A6.4 L1 data memory system

The L1 data cache is organized as a 4-way *Virtually Indexed, Physically Tagged* (VIPT) cache.

Data cache invalidate on reset

The Armv8-A architecture does not support an operation to invalidate the entire data cache. If software requires this function, it must be constructed by iterating over the cache geometry and executing a series of individual invalidate by set/way instructions.

A6.4.1 Memory system implementation

This section describes the implementation of the L1 memory system.

Limited ordering regions

The core offers support for four limited ordering region descriptors, as introduced by the Armv8.1 Limited ordering regions.

Atomic instructions

The Cortex-A78C core supports the atomic instructions added in Armv8.1 architecture.

Atomic instructions to cacheable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides.

When an instruction hits in the L1 data cache in a unique state, then it is performed as a near atomic in the L1 memory system. If the atomic operation misses in the L1 cache, or the line is shared with another core, then the atomic is sent as a far atomic on the core CHI interface.

If the operation misses everywhere within the cluster, and the interconnect supports far atomics, then the atomic is passed on to the interconnect to perform the operation.

When the operation hits anywhere inside the cluster, or when an interconnect does not support atomics, the L3 memory system performs the atomic operation. If the line is not already there, it allocates the line into the L3 cache. This depends on whether the DSU is configured with an L3 cache.

Therefore, if software prefers that the atomic is performed as a near atomic, precede the atomic instruction with a `PLDW` or `PRFM PSTL1KEEP` instruction.

Alternatively, the CPUECTLR can be programmed such that different types of atomic instructions attempt to execute as a near atomic.

Finally, for load atomics, the CPUECTLR can be programmed such that these atomics will always be performed near, regardless of whether the line is original in the L1 memory system or not. One cache fill will be made on an atomic. If the cache line is lost before the atomic operation can be made, it will be sent as a far atomic.

The Cortex-A78C core supports atomics to device or Non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, it will result in an abort.

For more information on the CPUECTLR register, see *B2.44 CPUECTLR_EL1, CPU Extended Control Register, EL1* on page B2-205.

LDAPR instructions

The core supports Load acquire instructions adhering to the RCpc consistency semantic introduced in the Armv8.3 extensions for A profile. This is reflected in register ID_AA64ISAR1_EL1 where bits[23:20] are set to `0b0001` to indicate that the core supports `LDAPRB`, `LDAPRH`, and `LDAPR` instructions implemented in AArch64.

Transient memory region

The core has a specific behavior for memory regions that are marked as write-back cacheable and transient, as defined in the Armv8.0 architecture.

For any load or store 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.
- When the line is evicted from the L1 data cache, the transient hint is passed to the L2 cache so that the replacement policy will not attempt to retain the line. When the line is subsequently evicted from the L2 cache, it bypasses the next level cache entirely.

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 to memory that are designated as Write-Back are treated the same as loads to Transient memory.

A6.4.2 Internal exclusive monitor

The Cortex-A78C core L1 memory system has an internal 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.ERG defines the size of the tagged block as 16 words, one cache line.

Note

A load/store exclusive instruction is any one of the following:

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

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

A6.5 Data prefetching

This section describes the data prefetching behavior for the Cortex-A78C core.

Preload instructions

The Cortex-A78C core supports the AArch64 *Prefetch Memory* (`PRFM`) instructions and the AArch32 *Prefetch Data* (`PLD`) and *Preload Data With Intent To Write* (`PLDW`) instructions. These instructions signal to the memory system that memory accesses from a specified address are likely to occur soon. The memory system acts by taking actions that aim to reduce the latency of the memory access when they occur. `PRFM` instructions perform a lookup in the cache, and if they miss and are to a cacheable address, a linefill starts. However, the `PRFM` instruction retires when its linefill is started, rather than waiting for the linefill to complete. This enables other instructions to execute while the linefill continues in the background.

The *Preload Instruction* (`PLI`) memory system hint performs preloading in the L2 cache for cacheable accesses if they miss in both the L1 instruction cache and L2 cache. Instruction preloading is performed in the background.

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

Data prefetching and monitoring

The load-store unit includes a hardware prefetcher that is responsible for generating prefetches targeting both the L1 and the L2 cache. The load side prefetcher uses the virtual address to prefetch to both the L1 and L2 Cache. The store side prefetcher uses the physical address, and only prefetches to the L2 Cache.

The CPUECTLR register allows you to have some control over the prefetcher. See *B2.44 CPUECTLR_EL1, CPU Extended Control Register, EL1* on page B2-205 for more information on the control of the prefetcher.

Use the prefetch memory system instructions for data prefetching where short sequences or irregular pattern fetches are required.

Data cache zero

The Armv8-A architecture introduces a *Data Cache Zero by Virtual Address* (`DC ZVA`) instruction.

In the Cortex-A78C core, this instruction enables a block of 64 bytes in memory, aligned to 64 bytes in size, to be set to zero.

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

A6.6 Direct access to internal memory

The Cortex-A78C core provides a mechanism to read the internal memory that is used by the L1 caches, L2 cache, and *Translation Lookaside Buffer* (TLB) structures through IMPLEMENTATION DEFINED System registers. This functionality can be useful when debugging software or hardware issues.

When the core executes in AArch64 state, there are six read-only registers that are used to access the contents of the internal memory. The internal memory is selected by programming the IMPLEMENTATION DEFINED RAMINDEX register (using SYS #6, c15, c0, #0 instruction). These operations are available only in EL3. In all other modes, executing these instructions results in an Undefined Instruction exception. The data is read from read-only registers as shown in the following table.

Table A6-1 AArch64 registers used to access internal memory

Register name	Function	Access	Operation	Rd Data
IDATA0_EL3	Instruction Register 0	Read-only	`S3_6_c15_c0_0`	Data
IDATA1_EL3	Instruction Register 1	Read-only	`S3_6_c15_c0_1`	Data
IDATA2_EL3	Instruction Register 2	Read-only	`S3_6_c15_c0_2`	Data
DDATA0_EL3	Data Register 0	Read-only	`S3_6_c15_c1_0`	Data
DDATA1_EL3	Data Register 1	Read-only	`S3_6_c15_c1_1`	Data
DDATA2_EL3	Data Register 2	Read-only	`S3_6_c15_c1_2`	Data

A6.6.1 Encoding for L1 instruction cache tag, L1 instruction cache data, L1 BTB, L1 GHB, L1 BIM, L1 TLB instruction, and L0 macro-op cache data

The following tables show the encoding required to select a given cache line.

Table A6-2 L1 instruction cache tag location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x00`
[23:20]	RES0
[19:18]	Way
[17:14]	RES0
[13:6]	Index [13:6]
[5:0]	RES0

Table A6-3 L1 instruction cache data location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x01`
[23:20]	RES0
[19:18]	Way
[17:14]	RES0
[13:3]	Index [13:3]
[2:0]	RES0

Table A6-4 L1 BTB data location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x02`
[23:15]	RES0
[14:4]	Index [14:4]
[3:0]	RES0

Table A6-5 L1 GHB data location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x03`
[23:14]	RES0
[13:5]	Index [13:5]
[4:0]	RES0

Table A6-6 BIM data location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x05`
[23:13]	RES0
[12:4]	Index [12:4]
[3:0]	RES0

Table A6-7 L1 instruction TLB data location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x04`
[23:8]	RES0
[7:0]	TLB Entry (->32)

Table A6-8 L0 Macro-op cache data location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x06`
[23:10]	RES0
[9:0]	Index [9:0]

L1 instruction tag RAM

The following tables show the data that is returned from accessing the L1 instruction tag RAM.

Table A6-9 L1 instruction cache tag format for instruction register 0

Bit field	Description
[31]	Non-secure identifier for the physical address
[30:3]	Physical address [39:12]
[2:1]	Instruction state [1:0] **00** Invalid **01** T32 **10** A32 **11** A64
[0]	Parity

Table A6-10 L1 instruction cache tag format for instruction register 1

Bit field	Description
[63:0]	0

Table A6-11 L1 instruction cache tag format for instruction register 2

Bit field	Description
[63:0]	0

L1 instruction data RAM

The following tables show the data that is returned from accessing the L1 instruction data RAM.

Table A6-12 L1 instruction cache data format for instruction register 0

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

Table A6-13 L1 instruction cache data format for instruction register 1

Bit field	Description
[63:9]	0
[8]	Parity
[7:0]	Data [71:64]

Table A6-14 L1 instruction cache data format for instruction register 2

Bit field	Description
[63:0]	0

L1 BTB RAM

The following tables show the data that is returned from accessing the L1 BTB RAM.

Table A6-15 L1 BTB cache format for instruction register 0

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

Table A6-16 L1 BTB cache format for instruction register 1

Bit field	Description
[63:30]	0
[29:0]	Data [93:0]

Table A6-17 L1 BTB cache format for instruction register 2

Bit field	Description
[63:0]	0

L1 GHB RAM

The following tables show the data that is returned from accessing the L1 GHB RAM.

Table A6-18 L1 GHB cache format for instruction register 0

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

Table A6-19 L1 GHB cache format for instruction register 1

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

Table A6-20 L1 GHB cache format for instruction register 2

Bit field	Description
[63:0]	0

L1 BIM RAM

The following tables show the data that is returned from accessing the L1 BIM RAM.

Table A6-21 L1 BIM cache format for instruction register 0

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

Table A6-22 L1 BIM cache format for instruction register 1

Bit field	Description
[63:0]	0

Table A6-23 L1 BIM cache format for instruction register 2

Bit field	Description
[63:0]	0

L1 instruction TLB RAM

The following tables show the data that is returned from accessing the L1 instruction TLB RAM.

Table A6-24 L1 instruction TLB cache format for instruction register 0

Bit field	Description
[63:59]	Virtual address [16:12]
[58:57]	PBHA [1:0]
[56]	TLB attribute
[55:53]	Memory attributes: **000** Device nGnRnE **001** Device nGnRE **010** Device nGRE **011** Device GRE **100** Non-cacheable **101** Write-Back No-Allocate **110** Write-Back Transient **111** Write-Back Read-Allocate and Write-Allocate
[52:50]	Page size: **000** 4KB **001** 16KB **010** 64KB **011** 256KB **100** 2MB **101** 32MB **11x** RES0
[49:46]	TLB attribute
[45]	Outer-shared
[44]	Inner-shared
[43:39]	TLB attribute
[38:23]	ASID
[22:7]	VMID
[6:5]	Translation regime: **00** Secure EL1/EL0 **01** Secure EL3 **10** Non-secure EL1/EL0 **11** Non-secure EL2

Table A6-24 L1 instruction TLB cache format for instruction register 0 (continued)

Bit field	Description
[4:1]	TLB attribute
[0]	Valid

Table A6-25 L1 instruction TLB cache format for instruction register 1

Bit field	Description
[60]	Non-secure
[59:32]	Physical address [39:12]
[31:0]	Virtual address [48:17]

L0 macro-op RAM

The following tables show the data that is returned from accessing the L0 macro-op RAM.

Table A6-26 L0 Macro-op cache format for instruction register 0

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

Table A6-27 L0 Macro-op cache format for instruction register 1

Bit field	Description
[63:34]	0
[33:0]	Macro-op data [97:64]

Table A6-28 L0 Macro-op cache format for instruction register 2

Bit field	Description
[63:0]	0

A6.6.2 Encoding for L1 data cache tag, L1 data cache data, and L1 TLB data

The core data cache consists of a 4-way set-associative structure.

The encoding, which is set in `Rd` in the appropriate `MCR` instruction, used to locate the required cache data entry for tag, data, and TLB memory is shown in the following tables. It is similar for both the tag RAM, data RAM, and TLB access. Data RAM access includes an additional field to locate the appropriate doubleword in the cache line.

Tag RAM encoding includes an additional field to select which one of the two cache channels must be used to perform any access.

Table A6-29 L1 data cache tag location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x08`
[23:20]	RES0
[19:18]	Way

Table A6-29 L1 data cache tag location encoding (continued)

Bit fields of Rd	Description
[17:16]	Copy **00** Tag RAM associated with Pipe 0 **01** Tag RAM associated with Pipe 1 **10** Tag RAM associated with Pipe 2 **11** RES0
[15:14]	RES0
[13:6]	Index [13:6]
[5:0]	RES0

Table A6-30 L1 data cache data location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x09`
[23:20]	RES0
[19:18]	Way
[17:16]	BankSel
[15:14]	Unused
[13:6]	Index [13:6]
[5:0]	RES0

Table A6-31 L1 data TLB location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x0A`
[23:6]	RES0
[4:0]	TLB Entry (0->31)

Data cache reads return 64 bits of data in Data Register 0, Data Register 1, and Data Register 2. If cache protection is supported, Data Register 2 is used to report ECC information using the format shown in the following tables.

L1 data cache tag RAM with ECC

The following tables show the data that is returned from accessing the L1 data cache tag RAM with ECC.

Table A6-32 L1 data cache tag format with ECC for data register 0

Bit field	Description
[63:41]	0
[40:34]	ECC
[33]	Non-secure identifier for the physical address

Table A6-32 L1 data cache tag format with ECC for data register 0 (continued)

Bit field	Description
[32:5]	Physical address [39:12]
[4:3]	RES0
[2]	Transient/WBNA
[1:0]	MESI **00** Invalid **01** Shared **10** Exclusive **11** Modified with respect to the L2 cache

Table A6-33 L1 data cache tag format with ECC for data register 1

Bit field	Description
[63:0]	0

Table A6-34 L1 data cache tag format with ECC for data register 2

Bit field	Description
[63:0]	0

L1 data cache tag RAM without ECC

The following tables show the data that is returned from accessing the L1 data cache tag RAM without ECC.

Table A6-35 L1 data cache tag format without ECC for data register 0

Bit field	Description
[63:34]	0
[33]	Non-secure identifier for the physical address
[32:5]	Physical address [39:12]
[4:3]	RES0
[2]	Transient/WBNA
[1:0]	MESI **00** Invalid **01** Shared **10** Exclusive **11** Modified

Table A6-36 L1 data cache tag format without ECC for data register 1

Bit field	Description
[63:0]	0

Table A6-37 L1 data cache tag format without ECC for data register 2

Bit field	Description
[63:0]	0

L1 data cache data RAM with ECC

The following tables show the data that is returned from accessing the L1 data cache data RAM with ECC.

Table A6-38 L1 data cache data format with ECC for data register 0

Bit field	Description
[63:0]	Word1_data [31:0], Word0_data [31:0]

Table A6-39 L1 data cache data format with ECC for data register 1

Bit field	Description
[63:0]	Word3_data [31:0], Word2_data [31:0]

Table A6-40 L1 data cache data format with ECC for data register 2

Bit field	Description
[63:32]	0
[31:0]	Word3_poison, Word3_ecc [6:0], Word2_poison, Word2_ecc [6:0], Word1_poison, Word1_ecc [6:0], Word0_poison, Word0_ecc [6:0]

L1 data cache data RAM without ECC

The following tables show the data that is returned from accessing the L1 data cache data RAM without ECC.

Table A6-41 L1 data cache data format without ECC for data register 0

Bit field	Description
[63:0]	Word1_data [31:0], Word0_data [31:0]

Table A6-42 L1 data cache data format without ECC for data register 1

Bit field	Description
[63:0]	Word3_data [31:0], Word2_data [31:0]

Table A6-43 L1 data cache data format without ECC for data register 2

Bit field	Description
[63:0]	0

L1 data TLB RAM

The following tables show the data that is returned from accessing the L1 data TLB RAM.

Table A6-44 L1 data TLB cache format for data register 0

Bit field	Description
[63:62]	Virtual address [13:12]
[58]	Outer-shared
[57]	Inner-shared
[52:50]	Memory attributes: **000** Device nGnRnE **001** Device nGnRE **010** Device nGRE **011** Device GRE **100** Non-cacheable **101** Write-Back No-Allocate **110** Write-Back Transient **111** Write-Back Read-Allocate and Write-Allocate
[38:36]	Page size: **000** 4KB **001** 16KB **010** 64KB **011** 256KB **100** 2MB **101** RES0 **110** 512MB **111** RES0
[35]	Non-secure
[34:33]	Translation regime: **00** Secure EL1/EL0 **01** Secure EL3 **10** Non-secure EL1/EL0 **11** Non-secure EL2
[32:17]	ASID
[16:1]	VMID
[0]	Valid

Table A6-45 L1 data TLB cache format for data register 1

Bit field	Description
[63]	PBHA [0]
[62:35]	Physical address [39:12]
[34:0]	Virtual address[48:14]

A6.6.3 Encoding for the L2 unified cache

The following tables show the encoding required to select a given cache line.

Table A6-46 L2 cache tag location encoding for 256KB[b]

Bit fields of Rd	Description
[31:24]	RAMID = `0x10`
[23:21]	RES0
[20:18]	Way (0->7)
[17:15]	RES0
[14:13]	Index [14:13]
[12:10]	XOR(Index [12:10], Way [2:0])
[9:6]	Index [9:6]
[5:0]	RES0

Table A6-47 L2 cache tag location encoding for 512KB[c]

Bit fields of Rd	Description
[31:24]	RAMID = `0x10`
[23:21]	RES0
[20:18]	Way (0->7)
[17:16]	RES0
[15:12]	Index [15:12]
[11:9]	XOR(Index [11:9], Way [2:0])
[8:6]	Index [8:6]
[5:0]	RES0

Table A6-48 L2 cache data location encoding for 256KB[b]

Bit fields of Rd	Description
[31:24]	RAMID = `0x11`
[23:21]	RES0
[20:18]	Way (0->7)

[b] Index [14:7] = XOR(Physical Address [14:7], Physical Address [22:15]) Index [6] = Physical Address [6]
[c] Index [15:7] = XOR(Physical Address [15:7], Physical Address [24:16]) Index [6] = Physical Address [6]

Table A6-48 L2 cache data location encoding for 256KB[b] (continued)

Bit fields of Rd	Description
[17:15]	RES0
[14:13]	Index [14:13]
[12:10]	XOR(Index [12:10], Way [2:0])
[9:4]	Index [9:4]
[3:0]	RES0

Table A6-49 L2 cache data location encoding for 512KB[c]

Bit fields of Rd	Description
[31:24]	RAMID = `0x11`
[23:21]	RES0
[20:18]	Way (0->7)
[17:16]	RES0
[15:12]	XOR(Index [15:12], b1000)
[11:9]	XOR(Index [11:9], Way [2:0])
[8:4]	Index [8:4]
[3:0]	RES0

Table A6-50 L2 victim location encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x12`
[23:17]	RES0
[16:6]	Index [16:6]
[5:0]	RES0

L2 tag RAM when L2 is configured with a 256KB cache size

The following tables show the data that is returned from accessing the L2 tag RAM when L2 is configured with a 256KB cache size.

Table A6-51 L2 tag format with a 256KB L2 cache size for data register 0

Bit field	Description
[63:42]	0
[41:35]	ECC [6:0] if configured with ECC for a 256KB L2 cache size, otherwise 0
[34:33]	PBHA [1:0]
[32:8]	Physical address [39:15]
[7]	Non-secure identifier for the physical address

Table A6-51 L2 tag format with a 256KB L2 cache size for data register 0 (continued)

Bit field	Description
[6:5]	Virtual index [13:12]
[4]	Shareable
[3]	L1 data cache valid
[2:0]	L2 State **101** Modified **001** Exclusive **x11** Shared **xx0** Invalid

Table A6-52 L2 tag format with a 256KB L2 cache size for data register 1

Bit field	Description
[63:0]	0

Table A6-53 L2 tag format with a 256KB L2 cache size for data register 2

Bit field	Description
[63:0]	0

L2 tag RAM when L2 is configured with a 512KB cache size

The following tables show the data that is returned from accessing the L2 tag RAM when L2 is configured with a 512KB cache size.

Table A6-54 L2 tag format with a 512KB L2 cache size for data register 0

Bit field	Description
[63:41]	0
[40:34]	ECC [6:0] if configured with ECC for a 512KB L2 cache size, otherwise 0
[33:32]	PBHA [1:0]
[31:8]	Physical address [39:16]
[7]	Non-secure identifier for the physical address
[6:5]	Virtual index [13:12]
[4]	Shareable

Table A6-54 L2 tag format with a 512KB L2 cache size for data register 0 (continued)

Bit field	Description
[3]	L1 data cache valid
[2:0]	L2 State **101** Modified **001** Exclusive **x11** Shared **xx0** Invalid

Table A6-55 L2 tag format with a 512KB L2 cache size for data register 1

Bit field	Description
[63:0]	0

Table A6-56 L2 tag format with a 512KB L2 cache size for data register 2

Bit field	Description
[63:0]	0

L2 data RAM

The following tables show the data that is returned from accessing the L2 data RAM.

Table A6-57 L2 data format for data register 0

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

Table A6-58 L2 data format for data register 1

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

Table A6-59 L2 data format for data register 2

Bit field	Description
[63:16]	0
[15:8]	ECC for Data [127:64] if configured with ECC
[7:0]	ECC for Data [63:0] if configured with ECC

L2 victim RAM

The following tables show the data that is returned from accessing the L2 victim RAM.

Table A6-60 L2 victim format for data register 0

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

Table A6-61 L2 victim format for data register 1

Bit field	Description
[63:0]	0

Table A6-62 L2 victim format for data register 2

Bit field	Description
[63:0]	0

A6.6.4 Encoding for the L2 TLB

The following section describes the encoding for L2 TLB direct accesses.

The following table shows the encoding that is required to select a given TLB entry.

Table A6-63 L2 TLB encoding

Bit fields of Rd	Description
[31:24]	RAMID = `0x18`
[23:21]	RES0
[20:18]	Way `0b000` way0 `0b001` way1 `0b010` way2 `0b011` way3
[17:8]	RES0
[7:0]	Index

L2 TLB

The following tables show the data that is returned from accessing the L2 TLB.

Table A6-64 L2 TLB format for instruction register 0

Bit field	Description
[63]	Outer-shared
[62]	Inner-shared

Table A6-64 L2 TLB format for instruction register 0 (continued)

Bit field	Description
[61]	RES0
[60:58]	Memory attributes: `0b000` Device nGnRnE `0b001` Device nGnRE `0b010` Device nGRE `0b011` Device GRE `0b100` Non-cacheable `0b101` Write-Back No-Allocate `0b110` Write-Back Transient `0b111` Write-Back Read-Allocate and Write-Allocate
[57:54]	RES0
[53:20]	Physical address When bit[6] is 0: • [53:26] = PA[39:12] • [25:20] = Don't care When bit[6] is 1: For coalesced 4K page (IDATA0[19:17]=`0b001`): • [53:28] = PA[39:14] • [27:26] = PA[13:12] for page 3 (highest memory address) • [25:24] = PA[13:12] for page 2 • [23:22] = PA[13:12] for page 1 • [21:20] = PA[13:12] for page 0 (lowest memory address) For coalesced 16K page (IDATA0[19:17]=`0b010`): • [53:30] = PA[39:16] • [27:26] = PA[15:14] for page 3 (highest memory address) • [25:24] = PA[15:14] for page 2 • [23:22] = PA[15:14] for page 1 • [21:20] = PA[15:14] for page 0 (lowest memory address) For coalesced 64K page (IDATA0[19:17]=`0b011`): • [53:32] = PA[39:18] • [27:26] = PA[17:16] for page 3 (highest memory address) • [25:24] = PA[17:16] for page 2 • [23:22] = PA[17:16] for page 1 • [21:20] = PA[17:16] for page 0 (lowest memory address)

Table A6-64 L2 TLB format for instruction register 0 (continued)

Bit field	Description
[19:17]	Page size: `0b000` 4KB `0b001` 16KB `0b010` 64KB `0b011` 256KB `0b100` 2MB `0b101` 32MB `0b110` 512MB `0b111` 1GB
[16:7]	RES0
[6]	Indicates that the entry is coalesced and holds translations for up to four contiguous pages
[5:2]	This bit field contains the valid bits for four contiguous pages. If the entry is non-coalesced, then `0b0001` indicates a valid entry.
[1:0]	RES0

Table A6-65 L2 TLB format for instruction register 1

Bit field	Description
[63:62]	VMID [1:0]
[61:46]	ASID [15:0]
[45:44]	PBHA [1:0]
[43][d]	Walk cache entry

[d] when bit [43] of Instruction register 1 is set, indicating that this is a walk cache entry, the decoding provided in this table is not valid.

Table A6-65 L2 TLB format for instruction register 1 (continued)

Bit field	Description
[42:14]	Virtual address When bit[6] is 0: • VA[48:12] = {IDATA1[42:14], INDEX[7:0]} When bit[6] is 1: For coalesced 4K page (IDATA0[19:17]=`0b001`): • VA[48:14] = {IDATA1[42:16], INDEX[7:0]} • VA[13:12] = `0b11` for page 3 (highest memory address) • VA[13:12] = `0b10` for page 2 • VA[13:12] = `0b01` for page 1 • VA[13:12] = `0b00` for page 0 (lowest memory address) For coalesced 16K page (IDATA0[19:17]=`0b010`): • VA[48:16] = {IDATA1[42:18], INDEX[7:0]} • VA[15:14] = `0b11` for page 3 (highest memory address) • VA[15:14] = `0b10` for page 2 • VA[15:14] = `0b01` for page 1 • VA[15:14] = `0b00` for page 0 (lowest memory address) For coalesced 64K page (IDATA0[19:17] = `0b011`): • VA[48:18] = {IDATA1[42:20], INDEX[7:0]} • VA[17:16] = `0b11` for page 3 (highest memory address) • VA[17:16] = `0b10` for page 2 • VA[17:16] = `0b01` for page 1 • VA[17:16] = `0b00` for page 0 (lowest memory address)
[13]	RES0
[12]	Non-secure
[11:1]	RES0
[0]	Non-global

Table A6-66 L2 TLB format for instruction register 2

Bit field	Description
[63:16]	RES0
[15:14]	Translation regime: `0b00` Secure EL1 `0b01` EL3 `0b10` Non-secure EL1 `0b11` EL2
[13:0]	VMID[15:2]

Chapter A7
L2 memory system

This chapter describes the L2 memory system.

It contains the following sections:

A7.1 About the L2 memory system

The L2 memory subsystem consists of:

- An 8-way set associative L2 cache with a configurable size of 256KB or 512KB. Cache lines have a fixed length of 64 bytes.
- Optional ECC protection for all RAM structures except victim array.
- Strictly inclusive with L1 data cache. Weakly inclusive with L1 instruction cache.
- Configurable CHI interface to the *DynamicIQ Shared Unit* (DSU) or CHI compliant system with support for 128-bit and 256-bit data widths.
- Dynamic biased replacement policy.
- *Modified Exclusive Shared Invalid* (MESI) coherency.

A7.2 About the L2 cache

The integrated L2 cache is the Point of Unification for the Cortex-A78C core. It handles both instruction and data requests from the instruction side and data side of each core respectively.

When fetched from the system, instructions are allocated to the L2 cache and can be invalidated during maintenance operations.

Note

Caches in the core are invalidated automatically at reset deassertion unless the core power mode is initialized to Debug recovery mode. See the *Power management* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual* for more information.

A7.3 Support for memory types

The Cortex-A78C core simplifies the coherency logic by downgrading some memory types.

- Memory that is marked as both Inner Write-Back Cacheable and Outer Write-Back Cacheable is cached in the L1 data cache and the L2 cache.
- Memory that is marked Inner Write-Through is downgraded to Non-cacheable.
- Memory that is marked Outer Write-Through or Outer Non-cacheable is downgraded to Non-cacheable, even if the inner attributes are Write-Back cacheable.

The following table shows the transaction capabilities of the Cortex-A78C core. It lists the maximum possible values for read, write, DVM issuing, and snoop capabilities of the private L2 cache.

Table A7-1 Cortex-A78C transaction capabilities

Attribute	Value	Description
Write issuing capability	46/54/60	Maximum number of outstanding write transactions. Dependent on the configured TQ size. (48/56/62)
Read issuing capability	46/54/60	Maximum number of outstanding read transactions. Dependent on the configured TQ size. (48/56/62)
Snoop acceptance capability	29/33/36	Maximum number of outstanding snoops and stashes accepted. Dependent on the TQ size. (48/56/62)
DVM issuing capability	46/54/60	Maximum number of outstanding DVMOp transactions. Dependent on the configured TQ size. (48/56/62)

Chapter A8
Reliability, Availability, and Serviceability

This chapter describes the *Reliability, Availability, and Serviceability* (RAS) features implemented in the Cortex-A78C core.

It contains the following sections:

A8.1 Cache ECC and parity

The Cortex-A78C core implements the *Reliability, Availability, Serviceability* (RAS) extension to the Armv8-A architecture which provides mechanisms for standardized reporting of the errors generated by cache protection mechanisms.

When configured with core cache protection, the Cortex-A78C core can detect and correct a 1-bit error in any RAM and detect 2-bit errors in some RAMs.

Note

For information about SCU-L3 cache protection, see the *Implementation options* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

The RAS extension improves the system by reducing unplanned outages:

- Transient errors can be detected and corrected before they cause application or system failure.
- Failing components can be identified and replaced.
- Failure can be predicted ahead of time to allow replacement during planned maintenance.

Errors that are present but not detected are known as latent or undetected errors. A transaction carrying a latent error is corrupted. In a system with no error detection, all errors are latent errors and are silently propagated by components until either:

- They are masked and do not affect the outcome of the system. These are benign or false errors.
- They affect the service interface of the system and cause failure. These are silent data corruptions.

The severity of a failure can range from minor to catastrophic. In many systems, data or service loss is regarded as more of a minor failure than data corruption, as long as backup data is available.

The RAS extension focuses on errors that are produced from hardware faults, which fall into two main categories:

- Transient faults
- Persistent faults

The RAS extension describes data corruption faults, which mostly occur in memories and on data links. RAS concepts can also be used for the management of other types of physical faults found in systems, such as lock-step errors, thermal trip, and mechanical failure. The RAS extension provides a common programmers model and mechanisms for fault handling and error recovery.

A8.2 Cache protection behavior

The configuration of the *Reliability, Availability, and Serviceability* (RAS) extension that is implemented in the Cortex-A78C core includes cache protection.

Cache protection ensures that the Cortex-A78C core is protected against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Cortex-A78C core have the following types of cache protection:

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

Interleaved parity
One bit of parity is applicable to the even bits of the word, and one bit of parity is applicable to the odd bits of the word.

SECDED
Single Error Correct, Double Error Detect.

Table A8-1 Cache protection behavior on page A8-109 indicates which protection type is applied to each RAM.

The core can progress and remain functionally correct when there is a single bit error in any RAM.

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

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

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

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

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

Table A8-1 Cache protection behavior

RAM	Protection type	Protection granule	Correction behavior
L0 macro-op cache	SED	48 bits	The line that contains the error is invalidated from the macro-op cache and fetched again from the L1 instruction cache.
L1 instruction cache tag	1 parity bit	31 bits	The line that contains the error is invalidated from the L1 instruction cache and fetched again from the subsequent memory system.
L1 instruction cache data	SED	72 bits	The line that contains the error is invalidated from the L1 instruction cache and fetched again from the subsequent memory system.
L1 BTB	None	-	-
L1 GHB	None	-	-
L1 BIM	None	-	-

Table A8-1 Cache protection behavior (continued)

RAM	Protection type	Protection granule	Correction behavior
L1 data cache tag	SECDED	34 bits + 7 bits for ECC attached to the word.	The cache line that contains the error gets evicted, corrected in line, and refilled to the core.
L1 data cache data	SECDED	32 bits of data +1 poison bit + 7 bits for ECC attached to the word.	The cache line that contains the error gets evicted, corrected in line, and refilled to the core.
L1 Prefetch History Table (PHT)	None	-	-
MMU translation cache	2 interleaved parity bits	71 bits	Entry invalidated, new pagewalk started to refetch it.
MMU replacement policy	None	-	
L2 cache tag	SECDED	256KB L2 - 7 ECC bits for 35 tag bits 512KB L2 - 7 ECC bits for 34 tag bits	Tag is corrected inline.
L2 cache data	SECDED	8 ECC bits for 64 data bits	Data is corrected inline.
L2 victim	None	-	-
L2 TQ data	SECDED	8 ECC bits for 64 data bits	Data is corrected inline.

To ensure that progress is guaranteed even in case of hard error, the core returns corrected data to the core, and no cache access is required after data correction.

A8.3 Uncorrected errors and data poisoning

When an error is detected, the correction mechanism is triggered. However, if the error is a 2-bit error in a RAM protected by ECC, then the error is not correctable.

The behavior on an uncorrected error depends on the type of RAM.

Uncorrected error detected in a data RAM

When an uncorrected error is detected in a data RAM, the chunk of data with the error is marked as poisoned. This poison information is then transferred with the data and stored in the cache if the data is allocated into another cache. The poisoned information is stored per 64 bits of data, except in the L1 data cache where it is stored per 32 bits of data.

Uncorrected error detected in a tag RAM

When an uncorrected error is detected in a tag RAM, either the address or coherency state of the line is not known, and the corresponding data cannot be poisoned. In this case, the line is invalidated and an error recovery interrupt is generated to notify software that data has potentially been lost.

A8.4 RAS error types

This section describes the *Reliability, Availability, Serviceability* (RAS) error types that are introduced by the RAS extension and supported in the Cortex-A78C core.

When a component accesses memory, an error might be detected in that memory and then be corrected, deferred, or detected but silently propagated. The following table lists the types of RAS errors that are supported in the Cortex-A78C core.

Table A8-2 RAS error types supported in the Cortex-A78C core

RAS error type	Definition
Corrected	A *Corrected Error* (CE) is reported for a single-bit ECC error on any protected RAM.
Deferred	A *Deferred Error* (DE) is reported for a double-bit ECC error that affects the data RAM on either the L1 data cache or the L2 cache.
Uncorrected	An *Uncorrected Error* (UE) is reported for a double-bit ECC error that affects the tag RAM of either the L1 data cache or the L2 cache. An Uncorrected Error is also reported for External aborts received in response to a store, data cache maintenance, instruction cache maintenance, TLBI maintenance, or cache copyback of dirty data.

A8.5 Error Synchronization Barrier

The *Error Synchronization Barrier* (`ESB`) instruction synchronizes unrecoverable system errors.

In the Cortex-A78C core, the `ESB` instruction allows efficient isolation of errors:

- The `ESB` instruction does not wait for completion of accesses that cannot generate an asynchronous External abort. For example, if all External aborts are handled synchronously or it is known that no such accesses are outstanding.
- The `ESB` instruction does not order accesses and does not guarantee a pipeline flush.

All system errors must be synchronized by an `ESB` instruction, which guarantees the following:

- All system errors that are generated before the `ESB` instruction have pended a *System Error Interrupts* (SEI) exception.
- If a physical SEI is pended by or was pending before the `ESB` instruction executes, then:
  - It is taken before completion of the `ESB` instruction, if the physical SEI exception is unmasked at the current Exception level.
  - The pending SEI is cleared, the SEI status is recorded in DISR_EL1, and DISR_EL1.A is set to 1 if the physical SEI exception is masked at the current Exception level. It indicates that the SEI exception was generated before the `ESB` instruction by instructions that occur in program order.
- If a virtual SEI is pended by or was pending before the `ESB` instruction executes, then:
  - It is taken before completion of the `ESB` instruction, if the virtual SEI exception is unmasked.
  - The pending virtual SEI is cleared and the SEI status is recorded in VDISR_EL2 using the information provided by software in VSESR_EL2, if the virtual SEI exception is masked.

After the `ESB` instruction, one of the following scenarios occurs:

- SEIs pended by errors are taken and their status is recorded in ESR_ELn.
- SEIs pended by errors are deferred and their status is recorded in DISR_EL1 or VDISR_EL2.

This includes unrecoverable SEIs that are generated by instructions, translation table walks, and instruction fetches on the same core.

Note

DISR_EL1 can only be accessed at EL1 and above. If EL2 is implemented and HCR_EL2.AMO is set to 1, then reads and writes of DISR_EL1 at Non-secure EL1 access VDISR_EL2.

See the following registers:

- *B2.58 DISR_EL1, Deferred Interrupt Status Register, EL1* on page B2-238.
- *B2.74 HCR_EL2, Hypervisor Configuration Register, EL2* on page B2-256.
- *B2.124 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2* on page B2-337.

A8.6 Error recording

The component that detects an error is called a node. The Cortex-A78C core is a node that interacts with the *DynamIQ Shared Unit* (DSU) node. There is one record per node for the errors detected.

For more information on error recording generated by cache protection, see the *Arm® Reliability, Availability, and Serviceability (RAS) Specification, Armv8, for the Armv8-A architecture profile*. The following points apply specifically to the Cortex-A78C core:

- Error recording is only available when the core cache protection is implemented.
- In the Cortex-A78C core, any error that is detected is reported and recorded in the error record registers.
- There are two error records provided, which can be selected with the ERRSELR_EL1 register:
  - Record 0 is private to the core, and is updated on any error in the core RAMs including L1 caches, TLB, and L2 cache.
  - Record 1 records any error in the L3 and snoop filter RAMs and is shared between all cores in the cluster.
- The fault handling interrupt is generated on the **nFAULTIRQ[0]** pin for L3 and snoop filter errors, or on the **nFAULTIRQ[n+1]** pin for core *n* L1 and L2 errors.

Related references

B2.60 ERRSELR_EL1, Error Record Select Register, EL1 on page B2-241
B2.61 ERXADDR_EL1, Selected Error Record Address Register, EL1 on page B2-242
B2.62 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page B2-243
B2.63 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B2-244
B2.64 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page B2-245
B2.65 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1, EL1 on page B2-246
B2.66 ERXPFGCDNR_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B2-247
B2.67 ERXPFGCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B2-248
B2.68 ERXPFGFR_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B2-250
B2.69 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B2-251

A8.7 Error injection

The Cortex-A78C core supports fault injection for the purpose of testing fault handling software.

The core is programmable to inject an error for any of the possible error types (corrected error, deferred error, uncontainable error, and recoverable error) on a future memory access. When that access is performed, the core responds as if an error was detected on that access by asserting error interrupts, logging information in the error records, and taking aborts as appropriate for the type of error. Injecting an error will not affect the data in the RAM or the checking process itself. When a real error is detected on an access for which an injected error is programmed, the injected error will not prevent the core from handling the real error. The RAS register might log the injected error or the real error in this case.

To get the error injection to work:

- Program the Error Record Select Register (ERRSELR_EL1) to select Error record 0.
- Program the Error Record Control Register (ERR0CTLR) to enable error detection/recovery and fault detection.
- Program the Error Pseudo Fault Generation Control Register (ERR0PFGCTL) to allow error injection.

Note

Cacheable code must also be executed, which will cause Cacheable transactions that can be injected with errors.

The following table describes all the possible types of error that the core can encounter and therefore inject.

Table A8-3 Errors injected in the Cortex-A78C core

Error type	Description
Corrected	A *Corrected Error* (CE) is generated for a single-bit ECC error on L1 data caches and L2 caches, both on data and tag RAMs.
Deferred	A *Deferred Error* (DE) is generated for a double-bit ECC error on L1 data caches and L2 caches, but only on data RAM.
Uncontainable	An *Uncontainable Error* (UC) is generated for a double-bit ECC error on L1 data caches and L2 caches, but only on tag RAM.

The following table describes the registers that handle error injection in the Cortex-A78C core.

Table A8-4 Error injection registers

Register name	Description
ERR0PFGFR	The ERR Pseudo Fault Generation Feature register defines which errors can be injected.
ERR0PFGCTLR	The ERR Pseudo Fault Generation Control register controls the errors that are injected.
ERR0PFGCDNR	The ERR Pseudo Fault Generation Count Down register controls the fault injection timing.

Note

This mechanism simulates the corruption of any RAM but the data is not actually corrupted.

Related references

B3.7 ERR0PFGCDNR, Error Pseudo Fault Generation Count Down Register on page B3-356
B3.8 ERR0PFGCTLR, Error Pseudo Fault Generation Control Register on page B3-357

B3.9 ERR0PFGFR, Error Pseudo Fault Generation Feature Register on page B3-361

Chapter A9
Generic Interrupt Controller CPU interface

This chapter describes the Cortex-A78C core implementation of the Arm *Generic Interrupt Controller* (GIC) CPU interface.

It contains the following sections:

A9.1 About the Generic Interrupt Controller CPU interface

The Cortex-A78C core implements the GIC CPU interface as described in the Arm *Generic Interrupt Controller Architecture Specification.*

This interfaces with an external GICv3 or GICv4 distributor component within the cluster system and is a resource for supporting and managing interrupts. The GIC CPU interface hosts registers to mask, identify, and control states of interrupts forwarded to that core. Each core in the cluster system has a GIC CPU interface component and connects to a common external distributor component.

Note

This chapter describes only features that are specific to the Cortex-A78C core implementation. Additional information specific to the cluster can be found in the *Interfaces* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

The GICv4 architecture supports:

- Two Security states
- 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 GIC includes interrupt grouping functionality that supports:

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

This chapter describes only features that are specific to the Cortex-A78C core implementation.

Related references

Chapter B4 GIC registers on page B4-367

A9.2 Bypassing the CPU interface

The GIC CPU Interface is always implemented within the Cortex-A78C core.

However, you can disable it if you assert the GICCDISABLE signal HIGH at reset. If you disable the GIC CPU interface, the input pins nVIRQ and nVFIQ can be driven by an external GIC in the SoC. GIC system register access generates UNDEFINED instruction exceptions when the GICCDISABLE signal is HIGH.

If the GIC is enabled, the input pins nVIRQ and nVFIQ must be tied off to HIGH. This is because the internal GIC CPU interface generates the virtual interrupt signals to the cores. The nIRQ and nFIQ signals are controlled by software, therefore there is no requirement to tie them HIGH.

Chapter A10
Advanced SIMD and floating-point support

This chapter describes the Advanced SIMD and floating-point features and registers in the Cortex-A78C core. The unit in charge of handling the Advanced SIMD and floating-point features is also referred to as the data engine in this manual.

It contains the following sections:

A10.1 About the Advanced SIMD and floating-point support

The Cortex-A78C core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set and the Advanced SIMD and floating-point instructions in the A32 and T32 instruction sets.

The Cortex-A78C floating-point implementation:

- Does not generate floating-point exceptions.
- Implements all scalar operations in hardware with support for all combinations of:
  - Rounding modes
  - Flush-to-zero
  - Default *Not a Number* (NaN) modes

The Armv8-A architecture does not define a separate version number for its Advanced SIMD and floating-point support in the AArch64 Execution state because the instructions are always implicitly present.

A10.2 Accessing the feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch64 Execution state only.

The Cortex-A78C core only supports AArch32 in EL0, therefore none of the feature identification registers are accessible in the AArch32 Execution state.

You can access the feature identification registers in the AArch64 Execution state using the `MRS` instruction, for example:

```
MRS <Xt>, ID_AA64PFR0_EL1 ; Read ID_AA64PFR0_EL1 into Xt
MRS <Xt>, MVFR0_EL1       ; Read MVFR0_EL1 into Xt
MRS <Xt>, MVFR1_EL1       ; Read MVFR1_EL1 into Xt
MRS <Xt>, MVFR2_EL1       ; Read MVFR2_EL1 into Xt
```

Table A10-1 AArch64 Advanced SIMD and scalar floating-point feature identification registers

Register name	Description
ID_AA64PFR0_EL1	*B2.84 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1* on page B2-273
MVFR0_EL1	*B5.4 MVFR0_EL1, Media and VFP Feature Register 0, EL1* on page B5-407
MVFR1_EL1	*B5.5 MVFR1_EL1, Media and VFP Feature Register 1, EL1* on page B5-409
MVFR2_EL1	*B5.6 MVFR2_EL1, Media and VFP Feature Register 2, EL1* on page B5-411

Part B
Register descriptions

Chapter B1
AArch32 system registers

This chapter describes the system registers in the AArch32 state.

It contains the following section:

B1.1 AArch32 architectural system register summary

This chapter identifies the AArch32 architectural System registers implemented in the Cortex-A78C core.

The following table identifies the architecturally defined registers that are implemented in the Cortex-A78C core. For a description of these registers, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

For the registers listed in the following table, coproc==`0b1111`.

Table B1-1 Architecturally defined registers

Name	CRn	Opc1	CRm	Opc2	Width	Description
CNTFRQ	c14	0	c0	0	32	Timer Clock Ticks per Second
CNTP_CTL	c14	0	c2	1	32	Counter-timer Physical Timer Control register
CNTP_CVAL	-	2	c14	-	64	Counter-timer Physical Timer CompareValue register
CNTP_TVAL	c14	0	c2	0	32	Counter-timer Physical Timer TimerValue register
CNTPCT	-	0	c14	-	64	Counter-timer Physical Count register
CNTV_CTL	c14	0	c3	1	32	Counter-timer Virtual Timer Control register
CNTV_CVAL	-	3	c14	-	64	Counter-timer Virtual Timer CompareValue register
CNTV_TVAL	c14	0	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTVCT	-	1	c14	-	64	Counter-timer Virtual Count register
CP15ISB	c7	0	c5	4	32	Instruction Synchronization Barrier System instruction
CP15DSB	c7	0	c10	4	32	Data Synchronization Barrier System instruction
CP15DMB	c7	0	c10	5	32	Data Memory Barrier System instruction
DLR	c4	3	c5	1	32	Debug Link Register
DSPSR	c4	3	c5	0	32	Debug Saved Program Status Register
TPIDRURO	c13	0	c0	3	32	User Read-Only Thread ID Register
TPIDRURW	c13	0	c0	2	32	User Read/Write Thread ID Register

Chapter B2
AArch64 system registers

This chapter describes the system registers in the AArch64 state.

It contains the following sections:

B2.1 AArch64 registers

This chapter provides information about the AArch64 system registers with IMPLEMENTATION DEFINED bit fields and IMPLEMENTATION DEFINED registers associated with the core.

The chapter provides IMPLEMENTATION SPECIFIC information, for a complete description of the registers, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The chapter is presented as follows:

AArch64 architectural system register summary

This section identifies the AArch64 architectural system registers implemented in the Cortex-A78C core that have IMPLEMENTATION DEFINED bit fields. The register descriptions for these registers only contain information about the IMPLEMENTATION DEFINED bits.

AArch64 IMPLEMENTATION DEFINED register summary

This section identifies the AArch64 architectural registers implemented in the Cortex-A78C core that are IMPLEMENTATION DEFINED.

AArch64 registers by functional group

This section groups the IMPLEMENTATION DEFINED registers and architectural system registers with IMPLEMENTATION DEFINED bit fields, as identified previously, by function. It also provides reset details for key register types.

Register descriptions

The remainder of the chapter provides register descriptions of the IMPLEMENTATION DEFINED registers and architectural system registers with IMPLEMENTATION DEFINED bit fields, as identified previously. These are listed in alphabetic order.

B2.2 AArch64 architectural system register summary

This section describes the AArch64 architectural system registers implemented in the Cortex-A78C core.

The section contains two tables:

Registers with implementation defined bit fields

This table identifies the architecturally defined registers in Cortex-A78C that have IMPLEMENTATION DEFINED bit fields. The register descriptions for these registers only contain information about the IMPLEMENTATION DEFINED bits.

See *Table B2-1 Registers with implementation defined bit fields* on page B2-133.

Other architecturally defined registers

This table identifies the other architecturally defined registers that are implemented in the Cortex-A78C core. These registers are described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

See *Other architecturally defined registers* on page B2-137.

Table B2-1 Registers with implementation defined bit fields

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
ACTLR_EL1	3	c1	0	c0	1	64	*B2.5 ACTLR_EL1, Auxiliary Control Register, EL1* on page B2-151
ACTLR_EL2	3	c1	4	c0	1	64	*B2.6 ACTLR_EL2, Auxiliary Control Register, EL2* on page B2-152
ACTLR_EL3	3	c1	6	c0	1	64	*B2.7 ACTLR_EL3, Auxiliary Control Register, EL3* on page B2-155
AIDR_EL1	3	c0	1	c0	7	32	*B2.14 AIDR_EL1, Auxiliary ID Register, EL1* on page B2-163
AFSR0_EL1	3	c5	0	c1	0	32	*B2.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1* on page B2-157
AFSR0_EL2	3	c5	4	c1	0	32	*B2.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2* on page B2-158
AFSR0_EL3	3	c5	6	c1	0	32	*B2.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3* on page B2-159
AFSR1_EL1	3	c5	0	c1	1	32	*B2.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1* on page B2-160
AFSR1_EL2	3	c5	4	c1	1	32	*B2.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2* on page B2-161
AFSR1_EL3	3	c5	6	c1	1	32	*B2.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3* on page B2-162
AMAIR_EL1	3	c10	0	c3	0	64	*B2.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1* on page B2-164
AMAIR_EL2	3	c10	4	c3	0	64	*B2.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2* on page B2-165
AMAIR_EL3	3	c10	6	c3	0	64	*B2.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3* on page B2-166
CCSIDR_EL1	3	c0	1	c0	0	32	*B2.33 CCSIDR_EL1, Cache Size ID Register, EL1* on page B2-186

Table B2-1 Registers with implementation defined bit fields (continued)

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
CLIDR_EL1	3	c0	1	c0	1	64	*B2.34 CLIDR_EL1, Cache Level ID Register, EL1* on page B2-188
CPACR_EL1	3	c1	0	c0	2	32	*B2.35 CPACR_EL1, Architectural Feature Access Control Register, EL1* on page B2-190
CPTR_EL2	3	c1	4	c1	2	32	*B2.36 CPTR_EL2, Architectural Feature Trap Register, EL2* on page B2-191
CPTR_EL3	3	c1	6	c1	2	32	*B2.37 CPTR_EL3, Architectural Feature Trap Register, EL3* on page B2-192
CSSELR_EL1	3	c0	2	c0	0	32	*B2.55 CSSELR_EL1, Cache Size Selection Register, EL1* on page B2-234
CTR_EL0	3	c0	3	c0	1	32	*B2.56 CTR_EL0, Cache Type Register, EL0* on page B2-235
DISR_EL1	3	c12	0	c1	1	64	*B2.58 DISR_EL1, Deferred Interrupt Status Register, EL1* on page B2-238
ERRIDR_EL1	3	c5	0	c3	0	32	*B2.59 ERRIDR_EL1, Error ID Register, EL1* on page B2-240
ERRSELR_EL1	3	c5	0	c3	1	32	*B2.60 ERRSELR_EL1, Error Record Select Register, EL1* on page B2-241
ERXADDR_EL1	3	c5	0	c4	3	64	*B2.61 ERXADDR_EL1, Selected Error Record Address Register, EL1* on page B2-242
ERXCTLR_EL1	3	c5	0	c4	1	64	*B2.62 ERXCTLR_EL1, Selected Error Record Control Register, EL1* on page B2-243
ERXFR_EL1	3	c5	0	c4	0	64	*B2.63 ERXFR_EL1, Selected Error Record Feature Register, EL1* on page B2-244
ERXMISC0_EL1	3	c5	0	c5	0	64	*B2.64 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1* on page B2-245
ERXMISC1_EL1	3	c5	0	c5	1	64	*B2.65 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1, EL1* on page B2-246
ERXSTATUS_EL1	3	c5	0	c4	2	32	*B2.69 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1* on page B2-251
ESR_EL1	3	c5	0	c2	0	32	*B2.70 ESR_EL1, Exception Syndrome Register, EL1* on page B2-252
ESR_EL2	3	c5	4	c2	0	32	*B2.71 ESR_EL2, Exception Syndrome Register, EL2* on page B2-253
ESR_EL3	3	c5	6	c2	0	32	*B2.72 ESR_EL3, Exception Syndrome Register, EL3* on page B2-254
HACR_EL2	3	c1	4	c1	7	32	*B2.73 HACR_EL2, Hyp Auxiliary Configuration Register, EL2* on page B2-255
HCR_EL2	3	c1	4	c1	0	64	*B2.74 HCR_EL2, Hypervisor Configuration Register, EL2* on page B2-256
ID_AFR0_EL1	3	c0	0	c1	3	32	*B2.86 ID_AFR0_EL1, AArch32 Auxiliary Feature Register 0, EL1* on page B2-276

Table B2-1 Registers with implementation defined bit fields (continued)

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
ID_DFR0_EL1	3	c0	0	c1	2	32	*B2.87 ID_DFR0_EL1, AArch32 Debug Feature Register 0, EL1* on page B2-277
ID_ISAR0_EL1	3	c0	0	c2	0	32	*B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
ID_ISAR1_EL1	3	c0	0	c2	1	32	*B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
ID_ISAR2_EL1	3	c0	0	c2	2	32	*B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
ID_ISAR3_EL1	3	c0	0	c2	3	32	*B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
ID_ISAR4_EL1	3	c0	0	c2	4	32	*B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287
ID_ISAR5_EL1	3	c0	0	c2	5	32	*B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289
ID_ISAR6_EL1	3	c0	0	c2	7	32	*B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291
ID_MMFR0_EL1	3	c0	0	c1	4	32	*B2.95 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0, EL1* on page B2-292
ID_MMFR1_EL1	3	c0	0	c1	5	32	*B2.96 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1, EL1* on page B2-294
ID_MMFR2_EL1	3	c0	0	c1	6	32	*B2.97 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2, EL1* on page B2-296
ID_MMFR3_EL1	3	c0	0	c1	7	32	*B2.98 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3, EL1* on page B2-298
ID_MMFR4_EL1	3	c0	0	c2	6	32	*B2.99 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4, EL1* on page B2-300
ID_PFR0_EL1	3	c0	0	c1	0	32	*B2.100 ID_PFR0_EL1, AArch32 Processor Feature Register 0, EL1* on page B2-302
ID_PFR1_EL1	3	c0	0	c1	1	32	*B2.101 ID_PFR1_EL1, AArch32 Processor Feature Register 1, EL1* on page B2-304
ID_PFR2_EL1	3	c0	0	c3	4	32	*B2.102 ID_PFR2_EL1, AArch32 Processor Feature Register 2, EL1* on page B2-306
ID_AA64DFR0_EL1	3	c0	0	c5	0	64	*B2.77 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1* on page B2-260
ID_AA64ISAR0_EL1	3	c0	0	c6	0	64	*B2.79 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1* on page B2-263
ID_AA64ISAR1_EL1	3	c0	0	c6	1	64	*B2.80 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1* on page B2-265
ID_AA64MMFR0_EL1	3	c0	0	c7	0	64	*B2.81 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1* on page B2-267
ID_AA64MMFR1_EL1	3	c0	0	c7	1	64	*B2.82 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1* on page B2-269

Table B2-1 Registers with implementation defined bit fields (continued)

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
ID_AA64MMFR2_EL1	3	c0	0	c7	2	64	*B2.83 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1* on page B2-271
ID_AA64PFR0_EL1	3	c0	0	c4	0	64	*B2.84 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1* on page B2-273
IFSR32_EL2	3	c5	4	c0	1	32	
LORC_EL1	3	c10	0	c4	3	64	*B2.103 LORC_EL1, LORegion Control Register, EL1* on page B2-307
LORID_EL1	3	c10	0	c4	7	64	*B2.104 LORID_EL1, LORegion ID Register, EL1* on page B2-308
LORN_EL1	3	c10	0	c4	2	64	*B2.105 LORN_EL1, LORegion Number Register, EL1* on page B2-309
MDCR_EL3	3	c1	6	c3	1	32	*B2.106 MDCR_EL3, Monitor Debug Configuration Register, EL3* on page B2-310
MIDR_EL1	3	c0	0	c0	0	32	*B2.107 MIDR_EL1, Main ID Register, EL1* on page B2-313
MPIDR_EL1	3	c0	0	c0	5	64	*B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314
PAR_EL1	3	c7	0	c4	0	64	*B2.109 PAR_EL1, Physical Address Register, EL1* on page B2-316
RVBAR_EL3	3	c12	6	c0	1	64	*B2.112 RVBAR_EL3, Reset Vector Base Address Register, EL3* on page B2-319
REVIDR_EL1	3	c0	0	c0	6	32	*B2.110 REVIDR_EL1, Revision ID Register, EL1* on page B2-317
SCTLR_EL1	3	c1	0	c0	0	32	*B2.113 SCTLR_EL1, System Control Register, EL1* on page B2-320
SCTLR_EL2	3	c1	4	c0	0	32	*B2.114 SCTLR_EL2, System Control Register, EL2* on page B2-323
SCTLR_EL12	3	c1	5	c0	0	32	*B2.113 SCTLR_EL1, System Control Register, EL1* on page B2-320
SCTLR_EL3	3	c1	6	c0	0	32	*B2.115 SCTLR_EL3, System Control Register, EL3* on page B2-325
TCR_EL1	3	c2	0	c0	2	64	*B2.116 TCR_EL1, Translation Control Register, EL1* on page B2-327
TCR_EL2	3	c2	4	c0	2	64	*B2.117 TCR_EL2, Translation Control Register, EL2 Primes* on page B2-329
TCR_EL3	3	c2	6	c0	2	64	*B2.118 TCR_EL3, Translation Control Register, EL3* on page B2-330
TTBR0_EL1	3	c2	0	c0	0	64	*B2.119 TTBR0_EL1, Translation Table Base Register 0, EL1* on page B2-332
TTBR0_EL2	3	c2	4	c0	0	64	*B2.120 TTBR0_EL2, Translation Table Base Register 0, EL2* on page B2-333
TTBR0_EL3	3	c2	6	c0	0	64	*B2.121 TTBR0_EL3, Translation Table Base Register 0, EL3* on page B2-334

Table B2-1 Registers with implementation defined bit fields (continued)

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
TTBR1_EL1	3	c2	0	c0	1	64	*B2.122 TTBR1_EL1, Translation Table Base Register 1, EL1* on page B2-335
TTBR1_EL2	3	c2	4	c0	1	64	*B2.123 TTBR1_EL2, Translation Table Base Register 1, EL2* on page B2-336
VDISR_EL2	3	c12	4	c1	1	64	*B2.124 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2* on page B2-337
VSESR_EL2	3	c5	4	c2	3	64	*B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B2-339
VTCR_EL2	3	c2	4	c1	2	32	*B2.127 VTCR_EL2, Virtualization Translation Control Register, EL2* on page B2-340
VTTBR_EL2	3	c2	4	c1	0	64	*B2.128 VTTBR_EL2, Virtualization Translation Table Base Register, EL2* on page B2-341

Table B2-2 Other architecturally defined registers

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
AFSR0_EL12	3	c5	5	1	0	32	Auxiliary Fault Status Register 0
AFSR1_EL12	3	c5	5	1	1	32	Auxiliary Fault Status Register 1
AMAIR_EL12	3	c10	5	c3	0	64	Auxiliary Memory Attribute Indirection Register
CNTFRQ_EL0	3	c14	3	0	0	32	Counter-timer Frequency register
CNTHCTL_EL2	3	c14	4	c1	0	32	Counter-timer Hypervisor Control register
CNTHP_CTL_EL2	3	c14	4	c2	1	32	Counter-timer Hypervisor Physical Timer Control register
CNTHP_CVAL_EL2	3	c14	4	c2	2	64	Counter-timer Hyp Physical CompareValue register
CNTHP_TVAL_EL2	3	c14	4	c2	0	32	Counter-timer Hyp Physical Timer TimerValue register
CNTHV_CTL_EL2	3	c14	4	c3	1	32	Counter-timer Virtual Timer Control register
CNTHV_CVAL_EL2	3	c14	4	c3	2	64	Counter-timer Virtual Timer CompareValue register
CNTHV_TVAL_EL2	3	c14	4	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTKCTL_EL1	3	c14	0	c1	0	32	Counter-timer Kernel Control register
CNTKCTL_EL12	3	c14	5	c1	0	32	Counter-timer Kernel Control register
CNTP_CTL_EL0	3	c14	3	c2	1	32	Counter-timer Physical Timer Control register
CNTP_CTL_EL02	3	c14	5	c2	1	32	Counter-timer Physical Timer Control register
CNTP_CVAL_EL0	3	c14	3	c2	2	64	Counter-timer Physical Timer CompareValue register
CNTP_CVAL_EL02	3	c14	5	c2	2	64	Counter-timer Physical Timer CompareValue register
CNTP_TVAL_EL0	3	c14	3	c2	0	32	Counter-timer Physical Timer TimerValue register
CNTP_TVAL_EL02	3	c14	5	c2	0	32	Counter-timer Physical Timer TimerValue register
CNTPCT_EL0	3	c14	3	c0	1	64	Counter-timer Physical Count register
CNTPS_CTL_EL1	3	c14	7	c2	1	32	Counter-timer Physical Secure Timer Control register
CNTPS_CVAL_EL1	3	c14	7	c2	2	64	Counter-timer Physical Secure Timer CompareValue register

Table B2-2 Other architecturally defined registers (continued)

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
CNTPS_TVAL_EL1	3	c14	7	c2	0	32	Counter-timer Physical Secure Timer TimerValue register
CNTV_CTL_EL0	3	c14	3	c3	1	32	Counter-timer Virtual Timer Control register
CNTV_CTL_EL02	3	c14	5	c3	1	32	Counter-timer Virtual Timer Control register
CNTV_CVAL_EL0	3	c14	3	c3	2	64	Counter-timer Virtual Timer CompareValue register
CNTV_CVAL_EL02	3	c14	5	c3	2	64	Counter-timer Virtual Timer CompareValue register
CNTV_TVAL_EL0	3	c14	3	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTV_TVAL_EL02	3	c14	5	c3	0	32	Counter-timer Virtual Timer TimerValue register
CNTVCT_EL0	3	c14	3	c0	2	64	Counter-timer Virtual Count register
CNTVOFF_EL2	3	c14	4	c0	3	64	Counter-timer Virtual Offset register
CONTEXTIDR_EL1	3	c13	0	c0	1	32	Context ID Register (EL1)
CONTEXTIDR_EL12	3	c13	5	c0	1	32	Context ID Register (EL12)
CONTEXTIDR_EL2	3	c13	4	c0	1	32	Context ID Register (EL2)
CPACR_EL12	3	c1	5	c0	2	32	Architectural Feature Access Control Register
CPTR_EL3	3	c1	6	c1	2	32	Architectural Feature Trap Register (EL3)
DACR32_EL2	3	c3	4	c0	0	32	Domain Access Control Register
ESR_EL12	3	c5	5	c2	0	32	Exception Syndrome Register (EL12)
FAR_EL1	3	c6	0	c0	0	64	Fault Address Register (EL1)
FAR_EL12	3	c6	5	c0	0	64	Fault Address Register (EL12)
FAR_EL2	3	c6	4	c0	0	64	Fault Address Register (EL2)
FAR_EL3	3	c6	6	c0	0	64	Fault Address Register (EL3)
FPEXC32_EL2	3	c5	4	c3	0	32	Floating-point Exception Control register
HPFAR_EL2	3	c6	4	c0	4	64	Hypervisor IPA Fault Address Register
HSTR_EL2	3	c1	4	c1	3	32	Hypervisor System Trap Register
ID_AA64AFR0_EL1	3	c0	0	c5	4	64	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	c0	0	c5	5	64	AArch64 Auxiliary Feature Register 1
ID_AA64DFR1_EL1	3	c0	0	c5	1	64	AArch64 Debug Feature Register 1
ID_AA64PFR1_EL1	3	c0	0	c4	1	64	AArch64 Core Feature Register 1
ISR_EL1	3	c12	0	c1	0	32	Interrupt Status Register
LOREA_EL1	3	c10	0	c4	1	64	LORegion End Address Register
LORSA_EL1	3	c10	0	c4	0	64	LORegion Start Address Register
MAIR_EL1	3	c10	0	c2	0	64	Memory Attribute Indirection Register (EL1)
MAIR_EL12	3	c10	5	c2	0	64	Memory Attribute Indirection Register (EL12)
MAIR_EL2	3	c10	4	c2	0	64	Memory Attribute Indirection Register (EL2)
MAIR_EL3	3	c10	6	c2	0	64	Memory Attribute Indirection Register (EL3)
MDCR_EL2	3	c1	4	c1	1	32	Monitor Debug Configuration Register

Table B2-2 Other architecturally defined registers (continued)

Name	Op0	CRn	Op1	CRm	Op2	Width	Description
MVFR0_EL1	3	c0	0	c3	0	32	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	c0	0	c3	1	32	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	c0	0	c3	2	32	AArch32 Media and VFP Feature Register 2
RMR_EL3	3	c12	6	c0	2	32	Reset Management Register
SCR_EL3	3	c1	6	c1	0	32	Secure Configuration Register
SDER32_EL3	3	c1	6	c1	1	32	AArch32 Secure Debug Enable Register
TCR_EL12	3	c2	5	c0	2	64	Translation Control Register (EL12)
TPIDR_EL0	3	c13	3	c0	2	64	EL0 Read/Write Software Thread ID Register
TPIDR_EL1	3	c13	0	c0	4	64	EL1 Software Thread ID Register
TPIDR_EL2	3	c13	4	c0	2	64	EL2 Software Thread ID Register
TPIDR_EL3	3	c13	6	c0	2	64	EL3 Software Thread ID Register
TPIDRRO_EL0	3	c13	3	c0	3	64	EL0 Read-Only Software Thread ID Register
TTBR0_EL12	3	c2	5	c0	0	64	Translation Table Base Register 0 (EL12)
TTBR1_EL12	3	c2	5	c0	1	64	Translation Table Base Register 1 (EL12)
VBAR_EL1	3	c12	0	c0	0	64	Vector Base Address Register (EL1)
VBAR_EL12	3	c12	5	c0	0	64	Vector Base Address Register (EL12)
VBAR_EL2	3	c12	4	c0	0	64	Vector Base Address Register (EL2)
VBAR_EL3	3	c12	6	c0	0	64	Vector Base Address Register (EL3)
VMPIDR_EL2	3	c0	4	c0	5	64	Virtualization Multiprocessor ID Register
VPIDR_EL2	3	c0	4	c0	0	32	Virtualization Core ID Register

B2.3 AArch64 implementation defined register summary

This section describes the AArch64 registers in the Cortex-A78C core that are IMPLEMENTATION DEFINED.

The following tables list the AArch64 IMPLEMENTATION DEFINED registers, sorted by opcode.

Table B2-3 AArch64 implementation defined registers

Name	Copro	CRn	Op1	CRm	Op2	Width	Description
ATCR_EL1	3	c15	0	c7	0	32	*B2.28 ATCR_EL1, Auxiliary Translation Control Register, EL1* on page B2-177
ATCR_EL2	3	c15	4	c7	0	32	*B2.29 ATCR_EL2, Auxiliary Translation Control Register, EL2* on page B2-179
ATCR_EL12	3	c15	5	c7	0	32	*B2.30 ATCR_EL12 , Alias to Auxiliary Translation Control Register EL1* on page B2-181
ATCR_EL3	3	c15	6	c7	0	32	*B2.31 ATCR_EL3, Auxiliary Translation Control Register, EL3* on page B2-182
AVTCR_EL2	3	c15	4	c7	1	32	*B2.32 AVTCR_EL2, Auxiliary Virtualized Translation Control Register, EL2* on page B2-184
CPUACTLR_EL1	3	c15	0	c1	0	64	*B2.38 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1* on page B2-193
CPUACTLR2_EL1	3	c15	0	c1	1	64	*B2.39 CPUACTLR2_EL1, CPU Auxiliary Control Register 2, EL1* on page B2-195
CPUACTLR3_EL1	3	c15	0	c1	2	64	*B2.40 CPUACTLR3_EL1, CPU Auxiliary Control Register 3, EL1* on page B2-197
CPUACTLR5_EL1	3	c15	0	c9	0	64	*B2.41 CPUACTLR5_EL1, CPU Auxiliary Control Register 5, EL1* on page B2-199
CPUACTLR6_EL1	3	c15	0	c9	1	64	*B2.42 CPUACTLR6_EL1, CPU Auxiliary Control Register 6, EL1* on page B2-201
CPUCFR_EL1	3	c15	0	c0	0	32	*B2.43 CPUCFR_EL1, CPU Configuration Register, EL1* on page B2-203
CPUECTLR_EL1	3	c15	0	c1	4	64	*B2.44 CPUECTLR_EL1, CPU Extended Control Register, EL1* on page B2-205
CPUECTLR2_EL1	3	c15	0	c1	5	64	*B2.45 CPUECTLR2_EL1, CPU Extended Control Register2, EL1* on page B2-213
CPUPCR_EL3	3	c15	6	c8	1	64	*B2.46 CPUPCR_EL3, CPU Private Control Register, EL3* on page B2-215
CPUPMR_EL3	3	c15	6	c8	3	64	*B2.48 CPUPMR_EL3, CPU Private Mask Register, EL3* on page B2-219
CPUPOR_EL3	3	c15	6	c8	2	64	*B2.50 CPUPOR_EL3, CPU Private Operation Register, EL3* on page B2-223
CPUPPMCR_EL3	3	c15	6	c2	0	64	*B2.52 CPUPPMCR_EL3, CPU Power Performance Management Configuration Register, EL3* on page B2-227
CPUPSELR_EL3	3	c15	6	c8	0	32	*B2.53 CPUPSELR_EL3, CPU Private Selection Register, EL3* on page B2-229

Table B2-3 AArch64 implementation defined registers (continued)

Name	Copro	CRn	Op1	CRm	Op2	Width	Description
CPUPWRCTLR_EL1	3	c15	0	c2	7	32	*B2.54 CPUPWRCTLR_EL1, Power Control Register, EL1* on page B2-231
ERXPFGCDNR_EL1	3	c15	0	c2	2	32	*B2.66 ERXPFGCDNR_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1* on page B2-247
ERXPFGCTLR_EL1	3	c15	0	c2	1	32	*B2.67 ERXPFGCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1* on page B2-248
ERXPFGFR_EL1	3	c15	0	c2	0	32	*B2.68 ERXPFGFR_EL1, Selected Pseudo Fault Generation Feature Register, EL1* on page B2-250

The following table shows the 32-bit wide IMPLEMENTATION DEFINED Cluster registers. Details of these registers can be found in *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*

Table B2-4 Cluster registers

Name	Copro	CRn	Opc1	CRm	Opc2	Width	Description
CLUSTERCFR_EL1	3	c15	0	c3	0	32-bit	Cluster configuration register
CLUSTERIDR_EL1	3	c15	0	c3	1	32-bit	Cluster main revision ID
CLUSTEREVIDR_EL1	3	c15	0	c3	2	32-bit	Cluster ECO ID
CLUSTERACTLR_EL1	3	c15	0	c3	3	32-bit	Cluster auxiliary control register
CLUSTERECTLR_EL1	3	c15	0	c3	4	32-bit	Cluster extended control register
CLUSTERPWRCTLR_EL1	3	c15	0	c3	5	32-bit	Cluster power control register
CLUSTERPWRDN_EL1	3	c15	0	c3	6	32-bit	Cluster power down register
CLUSTERPWRSTAT_EL1	3	c15	0	c3	7	32-bit	Cluster power status register
CLUSTERTHREADSID_EL1	3	c15	0	c4	0	32-bit	Cluster thread scheme ID register
CLUSTERACPSID_EL1	3	c15	0	c4	1	32-bit	Cluster ACP scheme ID register
CLUSTERSTASHSID_EL1	3	c15	0	c4	2	32-bit	Cluster stash scheme ID register
CLUSTERPARTCR_EL1	3	c15	0	c4	3	32-bit	Cluster partition control register
CLUSTERBUSQOS_EL1	3	c15	0	c4	4	32-bit	Cluster bus QoS control register
CLUSTERL3HIT_EL1	3	c15	0	c4	5	32-bit	Cluster L3 hit counter register
CLUSTERL3MISS_EL1	3	c15	0	c4	6	32-bit	Cluster L3 miss counter register
CLUSTERTHREADSIDOVR_EL1	3	c15	0	c4	7	32-bit	Cluster thread scheme ID override register
CLUSTERPMCR_EL1	3	c15	0	c5	0	32-bit	Cluster Performance Monitors Control Register
CLUSTERPMCNTENSET_EL1	3	c15	0	c5	1	32-bit	Cluster Count Enable Set Register
CLUSTERPMCNTENCLR_EL1	3	c15	0	c5	2	32-bit	Cluster Count Enable Clear Register
CLUSTERPMOVSSET_EL1	3	c15	0	c5	3	32-bit	Cluster Overflow Flag Status Set
CLUSTERPMOVSCLR_EL1	3	c15	0	c5	4	32-bit	Cluster Overflow Flag Status Clear
CLUSTERPMSELR_EL1	3	c15	0	c5	5	32-bit	Cluster Event Counter Selection Register
CLUSTERPMINTENSET_EL1	3	c15	0	c5	6	32-bit	Cluster Interrupt Enable Set Register

Table B2-4 Cluster registers (continued)

Name	Copro	CRn	Opc1	CRm	Opc2	Width	Description
CLUSTERPMINTENCLR_EL1	3	c15	0	c5	7	32-bit	Cluster Interrupt Enable Clear Register
CLUSTERPMXEVTYPER_EL1	3	c15	0	c6	1	32-bit	Cluster Selected Event Type and Filter Register
CLUSTERPMXEVCNTR_EL1	3	c15	0	c6	2	32-bit	Cluster Selected Event Counter Register
Reserved/RAZ	3	c15	0	c6	3	32-bit	Cluster Monitor Debug Configuration Register
CLUSTERPMCEID0_EL1	3	c15	0	c6	4	32-bit	Cluster Common Event Identification ID0 Register
CLUSTERPMCEID1_EL1	3	c15	0	c6	5	32-bit	Cluster Common Event Identification ID1 Register
CLUSTERPMCLAIMSET_EL1	3	c15	0	c6	6	32-bit	Cluster Performance Monitor Claim Tag Set Register
CLUSTERPMCLAIMCLR_EL1	3	c15	0	c6	7	32-bit	Cluster Performance Monitor Claim Tag Clear Register

B2.4 AArch64 registers by functional group

This section identifies the AArch64 registers by their functional groups and applies to the registers in the core that are IMPLEMENTATION DEFINED or have micro-architectural bit fields. Reset values are provided for these registers.

Identification registers

Name	Type	Reset	Description
AIDR_EL1	RO	`0x00000000`	*B2.14 AIDR_EL1, Auxiliary ID Register, EL1* on page B2-163
CCSIDR_EL1	RO	-	*B2.33 CCSIDR_EL1, Cache Size ID Register, EL1* on page B2-186
CLIDR_EL1	RO	• `0xC3000123` if L3 cache present. • `0x82000023` if no L3 cache.	*B2.34 CLIDR_EL1, Cache Level ID Register, EL1* on page B2-188
CSSELR_EL1	RW	UNK	*B2.55 CSSELR_EL1, Cache Size Selection Register, EL1* on page B2-234
CTR_EL0	RO	`0x9444C004`	*B2.56 CTR_EL0, Cache Type Register, EL0* on page B2-235
DCZID_EL0	RO	`0x00000004`	*B2.57 DCZID_EL0, Data Cache Zero ID Register, EL0* on page B2-237
ERRIDR_EL1	RO	`0x00000002`	*B2.59 ERRIDR_EL1, Error ID Register, EL1* on page B2-240
ID_AA64AFR0_EL1	RO	`0x00000000`	*B2.75 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0* on page B2-258
ID_AA64AFR1_EL1	RO	`0x00000000`	*B2.76 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1* on page B2-259
ID_AA64DFR0_EL1	RO	`0x0000000010305408`	*B2.77 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1* on page B2-260
ID_AA64DFR1_EL1	RO	`0x00000000`	*B2.78 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1* on page B2-262
ID_AA64ISAR0_EL1	RO	• `0x0010100010211120` if the Cryptographic Extension is implemented. • `0x0010100010210000` if the Cryptographic Extension is not implemented.	*B2.79 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1* on page B2-263
ID_AA64ISAR1_EL1	RO	`0x0000000001200031`	*B2.80 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1* on page B2-265
ID_AA64MMFR0_EL1	RO	`0x0000000000101122`	*B2.81 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1* on page B2-267
ID_AA64MMFR1_EL1	RO	`0x0000000010212122`	*B2.82 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1* on page B2-269

(continued)

Name	Type	Reset	Description
ID_AA64MMFR2_EL1	RO	0x0000000100001011	*B2.83 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1* on page B2-271
ID_AA64PFR0_EL1	RO	• 0x1100000010111112 if the GICv4 interface is disabled. • 0x1100000011111112 if the GICv4 interface is enabled.	*B2.84 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1* on page B2-273
ID_AA64PFR1_EL1	RO	0x0000000000000010	*B2.85 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1* on page B2-275
ID_AFR0_EL1	RO	0x00000000	*B2.86 ID_AFR0_EL1, AArch32 Auxiliary Feature Register 0, EL1* on page B2-276
ID_DFR0_EL1	RO	0x04010088	*B2.87 ID_DFR0_EL1, AArch32 Debug Feature Register 0, EL1* on page B2-277
ID_ISAR0_EL1	RO	0x02101110	*B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
ID_ISAR1_EL1	RO	0x13112111	*B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
ID_ISAR2_EL1	RO	0x21232042	*B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
ID_ISAR3_EL1	RO	0x01112131	*B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
ID_ISAR4_EL1	RO	0x00010142	*B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287
ID_ISAR5_EL1	RO	0x01011121 ID_ISAR5 has the value 0x01010001 if the Cryptographic Extension is not implemented and enabled.	*B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289
ID_ISAR6_EL1	RO	0x00000010	*B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291
ID_MMFR0_EL1	RO	0x10201105	*B2.95 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0, EL1* on page B2-292
ID_MMFR1_EL1	RO	0x40000000	*B2.96 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1, EL1* on page B2-294
ID_MMFR2_EL1	RO	0x01260000	*B2.97 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2, EL1* on page B2-296
ID_MMFR3_EL1	RO	0x02122211	*B2.98 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3, EL1* on page B2-298

(continued)

Name	Type	Reset	Description
ID_MMFR4_EL1	RO	`0x00021110`	*B2.99 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4, EL1* on page B2-300
ID_PFR0_EL1	RO	`0x10010131`	*B2.100 ID_PFR0_EL1, AArch32 Processor Feature Register 0, EL1* on page B2-302
ID_PFR1_EL1	RO	`0x10010000` Bits [31:28] are `0x1` if the GIC CPU interface is implemented and enabled, and `0x0` otherwise.	*B2.101 ID_PFR1_EL1, AArch32 Processor Feature Register 1, EL1* on page B2-304
LORID_EL1	RO	`0x0000000000040004`	*B2.104 LORID_EL1, LORegion ID Register, EL1* on page B2-308
MIDR_EL1	RO	`0x411FD410`	*B2.107 MIDR_EL1, Main ID Register, EL1* on page B2-313
MPIDR_EL1	RO	The reset value depends on **CLUSTERIDAFF2[7:0]** and **CLUSTERIDAFF3[7:0]**. See register description for details.	*B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314
REVIDR_EL1	RO	`0x00000000`	*B2.110 REVIDR_EL1, Revision ID Register, EL1* on page B2-317
VMPIDR_EL2	RW	The reset value is the value of MPIDR_EL1.	Virtualization Multiprocessor ID Register EL2
VPIDR_EL2	RW	The reset value is the value of MIDR_EL1.	Virtualization Core ID Register EL2

Other system control registers

Name	Type	Description
ACTLR_EL1	RW	*B2.5 ACTLR_EL1, Auxiliary Control Register, EL1* on page B2-151
ACTLR_EL2	RW	*B2.6 ACTLR_EL2, Auxiliary Control Register, EL2* on page B2-152
ACTLR_EL3	RW	*B2.7 ACTLR_EL3, Auxiliary Control Register, EL3* on page B2-155
CPACR_EL1	RW	*B2.35 CPACR_EL1, Architectural Feature Access Control Register, EL1* on page B2-190
SCTLR_EL1	RW	*B2.113 SCTLR_EL1, System Control Register, EL1* on page B2-320
SCTLR_EL2	RW	*B2.114 SCTLR_EL2, System Control Register, EL2* on page B2-323
SCTLR_EL3	RW	*B2.115 SCTLR_EL3, System Control Register, EL3* on page B2-325
SCTLR_EL12	RW	*B2.113 SCTLR_EL1, System Control Register, EL1* on page B2-320

Reliability, Availability, Serviceability (RAS) registers

Name	Type	Description
DISR_EL1	RW	*B2.58 DISR_EL1, Deferred Interrupt Status Register, EL1* on page B2-238
ERRIDR_EL1	RW	*B2.59 ERRIDR_EL1, Error ID Register, EL1* on page B2-240
ERRSELR_EL1	RW	*B2.60 ERRSELR_EL1, Error Record Select Register, EL1* on page B2-241
ERXADDR_EL1	RW	*B2.61 ERXADDR_EL1, Selected Error Record Address Register, EL1* on page B2-242
ERXCTLR_EL1	RW	*B2.62 ERXCTLR_EL1, Selected Error Record Control Register, EL1* on page B2-243
ERXFR_EL1	RO	*B2.63 ERXFR_EL1, Selected Error Record Feature Register, EL1* on page B2-244
ERXMISC0_EL1	RW	*B2.64 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1* on page B2-245
ERXMISC1_EL1	RW	*B2.65 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1, EL1* on page B2-246
ERXSTATUS_EL1	RW	*B2.69 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1* on page B2-251
ERXPFGCDNR_EL1	RW	*B2.66 ERXPFGCDNR_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1* on page B2-247
ERXPFCTLR_EL1	RW	*B2.67 ERXPFGCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1* on page B2-248
ERXPFGFR_EL1	RO	*B2.68 ERXPFGFR_EL1, Selected Pseudo Fault Generation Feature Register, EL1* on page B2-250
HCR_EL2	RW	*B2.74 HCR_EL2, Hypervisor Configuration Register, EL2* on page B2-256
VDISR_EL2	RW	*B2.124 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2* on page B2-337
VSESR_EL2	RW	*B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B2-339

Virtual Memory control registers

Name	Type	Description
AMAIR_EL1	RW	*B2.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1* on page B2-164
AMAIR_EL2	RW	*B2.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2* on page B2-165
AMAIR_EL3	RW	*B2.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3* on page B2-166
ATCR_EL1	RW	*B2.28 ATCR_EL1, Auxiliary Translation Control Register, EL1* on page B2-177
ATCR_EL2	RW	*B2.29 ATCR_EL2, Auxiliary Translation Control Register, EL2* on page B2-179
ATCR_EL12	RW	*B2.30 ATCR_EL12 , Alias to Auxiliary Translation Control Register EL1* on page B2-181
ATCR_EL3	RW	*B2.31 ATCR_EL3, Auxiliary Translation Control Register, EL3* on page B2-182
AVTCR_EL2	RW	*B2.32 AVTCR_EL2, Auxiliary Virtualized Translation Control Register, EL2* on page B2-184
LORC_EL1	RW	*B2.103 LORC_EL1, LORegion Control Register, EL1* on page B2-307
LOREA_EL1	RW	LORegion End Address Register EL1

(continued)

Name	Type	Description
LORID_EL1	RO	*B2.104 LORID_EL1, LORegion ID Register, EL1* on page B2-308
LORN_EL1	RW	*B2.105 LORN_EL1, LORegion Number Register, EL1* on page B2-309
LORSA_EL1	RW	LORegion Start Address Register EL1
TCR_EL1	RW	*B2.116 TCR_EL1, Translation Control Register, EL1* on page B2-327
TCR_EL2	RW	*B2.117 TCR_EL2, Translation Control Register, EL2 Primes* on page B2-329
TCR_EL3	RW	*B2.118 TCR_EL3, Translation Control Register, EL3* on page B2-330
TTBR0_EL1	RW	*B2.119 TTBR0_EL1, Translation Table Base Register 0, EL1* on page B2-332
TTBR0_EL2	RW	*B2.120 TTBR0_EL2, Translation Table Base Register 0, EL2* on page B2-333
TTBR0_EL3	RW	*B2.121 TTBR0_EL3, Translation Table Base Register 0, EL3* on page B2-334
TTBR1_EL1	RW	*B2.122 TTBR1_EL1, Translation Table Base Register 1, EL1* on page B2-335
TTBR1_EL2	RW	*B2.123 TTBR1_EL2, Translation Table Base Register 1, EL2* on page B2-336
VTTBR_EL2	RW	*B2.128 VTTBR_EL2, Virtualization Translation Table Base Register, EL2* on page B2-341

Virtualization registers

Name	Type	Description
ACTLR_EL2	RW	*B2.6 ACTLR_EL2, Auxiliary Control Register, EL2* on page B2-152
AFSR0_EL2	RW	*B2.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2* on page B2-158
AFSR1_EL2	RW	*B2.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2* on page B2-161
AMAIR_EL2	RW	*B2.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2* on page B2-165
CPTR_EL2	RW	*B2.36 CPTR_EL2, Architectural Feature Trap Register, EL2* on page B2-191
ESR_EL2	RW	*B2.71 ESR_EL2, Exception Syndrome Register, EL2* on page B2-253
HACR_EL2	RW	*B2.73 HACR_EL2, Hyp Auxiliary Configuration Register, EL2* on page B2-255
HCR_EL2	RW	*B2.74 HCR_EL2, Hypervisor Configuration Register, EL2* on page B2-256
HPFAR_EL2	RW	Hypervisor IPA Fault Address Register EL2
TCR_EL2	RW	*B2.117 TCR_EL2, Translation Control Register, EL2 Primes* on page B2-329
VMPIDR_EL2	RW	Virtualization Multiprocessor ID Register EL2
VPIDR_EL2	RW	Virtualization Core ID Register EL2
VSESR_EL2	RW	*B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B2-339

(continued)

Name	Type	Description
VTCR_EL2	RW	*B2.127 VTCR_EL2, Virtualization Translation Control Register, EL2* on page B2-340
VTTBR_EL2	RW	*B2.128 VTTBR_EL2, Virtualization Translation Table Base Register, EL2* on page B2-341

Exception and fault handling registers

Name	Type	Description
AFSR0_EL1	RW	*B2.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1* on page B2-157
AFSR0_EL2	RW	*B2.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2* on page B2-158
AFSR0_EL3	RW	*B2.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3* on page B2-159
AFSR1_EL1	RW	*B2.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1* on page B2-160
AFSR1_EL2	RW	*B2.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2* on page B2-161
AFSR1_EL3	RW	*B2.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3* on page B2-162
DISR_EL1	RW	*B2.58 DISR_EL1, Deferred Interrupt Status Register, EL1* on page B2-238
ESR_EL1	RW	*B2.70 ESR_EL1, Exception Syndrome Register, EL1* on page B2-252
ESR_EL2	RW	*B2.71 ESR_EL2, Exception Syndrome Register, EL2* on page B2-253
ESR_EL3	RW	*B2.72 ESR_EL3, Exception Syndrome Register, EL3* on page B2-254
HPFAR_EL2	RW	Hypervisor IPA Fault Address Register EL2
VDISR_EL2	RW	*B2.124 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2* on page B2-337
VSESR_EL2	RW	*B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B2-339

Implementation defined registers

Name	Type	Description
ATCR_EL1	RW	*B2.28 ATCR_EL1, Auxiliary Translation Control Register, EL1* on page B2-177
ATCR_EL2	RW	*B2.29 ATCR_EL2, Auxiliary Translation Control Register, EL2* on page B2-179
ATCR_EL12	RW	*B2.30 ATCR_EL12 , Alias to Auxiliary Translation Control Register EL1* on page B2-181
ATCR_EL3	RW	*B2.31 ATCR_EL3, Auxiliary Translation Control Register, EL3* on page B2-182
AVTCR_EL2	RW	*B2.32 AVTCR_EL2, Auxiliary Virtualized Translation Control Register, EL2* on page B2-184
CPUACTLR_EL1	RW	*B2.38 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1* on page B2-193
CPUACTLR2_EL1	RW	*B2.39 CPUACTLR2_EL1, CPU Auxiliary Control Register 2, EL1* on page B2-195
CPUACTLR3_EL1	RW	*B2.40 CPUACTLR3_EL1, CPU Auxiliary Control Register 3, EL1* on page B2-197
CPUACTLR5_EL1	RW	*B2.41 CPUACTLR5_EL1, CPU Auxiliary Control Register 5, EL1* on page B2-199

(continued)

Name	Type	Description
CPUACTLR6_EL1	RW	*B2.42 CPUACTLR6_EL1, CPU Auxiliary Control Register 6, EL1* on page B2-201
CPUCFR_EL1	RO	*B2.43 CPUCFR_EL1, CPU Configuration Register, EL1* on page B2-203
CPUECTLR_EL1	RW	*B2.44 CPUECTLR_EL1, CPU Extended Control Register, EL1* on page B2-205
CPUECTLR2_EL1	RW	*B2.45 CPUECTLR2_EL1, CPU Extended Control Register2, EL1* on page B2-213
CPUPWRCTLR_EL1	RW	*B2.54 CPUPWRCTLR_EL1, Power Control Register, EL1* on page B2-231
ERXPFGCDNR_EL1	RW	*B2.66 ERXPFGCDNR_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1* on page B2-247
ERXPFGCTLR_EL1	RW	*B2.67 ERXPFGCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1* on page B2-248
ERXPFGFR_EL1	RW	*B2.68 ERXPFGFR_EL1, Selected Pseudo Fault Generation Feature Register, EL1* on page B2-250

The following table shows the 32-bit wide IMPLEMENTATION DEFINED Cluster registers. Details of these registers can be found in *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*

Table B2-5 Cluster registers

Name	Copro	CRn	Opc1	CRm	Opc2	Width	Description
CLUSTERCFR_EL1	3	c15	0	c3	0	32-bit	Cluster configuration register.
CLUSTERIDR_EL1	3	c15	0	c3	1	32-bit	Cluster main revision ID.
CLUSTEREVIDR_EL1	3	c15	0	c3	2	32-bit	Cluster ECO ID.
CLUSTERACTLR_EL1	3	c15	0	c3	3	32-bit	Cluster auxiliary control register.
CLUSTERECTLR_EL1	3	c15	0	c3	4	32-bit	Cluster extended control register.
CLUSTERPWRCTLR_EL1	3	c15	0	c3	5	32-bit	Cluster power control register.
CLUSTERPWRDN_EL1	3	c15	0	c3	6	32-bit	Cluster power down register.
CLUSTERPWRSTAT_EL1	3	c15	0	c3	7	32-bit	Cluster power status register.
CLUSTERTHREADSID_EL1	3	c15	0	c4	0	32-bit	Cluster thread scheme ID register.
CLUSTERACPSID_EL1	3	c15	0	c4	1	32-bit	Cluster ACP scheme ID register.
CLUSTERSTASHSID_EL1	3	c15	0	c4	2	32-bit	Cluster stash scheme ID register.
CLUSTERPARTCR_EL1	3	c15	0	c4	3	32-bit	Cluster partition control register.
CLUSTERBUSQOS_EL1	3	c15	0	c4	4	32-bit	Cluster bus QoS control register.
CLUSTERL3HIT_EL1	3	c15	0	c4	5	32-bit	Cluster L3 hit counter register.
CLUSTERL3MISS_EL1	3	c15	0	c4	6	32-bit	Cluster L3 miss counter register.
CLUSTERTHREADSIDOVR_EL1	3	c15	0	c4	7	32-bit	Cluster thread scheme ID override register
CLUSTERPMCR_EL1	3	c15	0	c5	0	32-bit	Cluster Performance Monitors Control Register
CLUSTERPMCNTENSET_EL1	3	c15	0	c5	1	32-bit	Cluster Count Enable Set Register
CLUSTERPMCNTENCLR_EL1	3	c15	0	c5	2	32-bit	Cluster Count Enable Clear Register

Table B2-5 Cluster registers (continued)

Name	Copro	CRn	Opc1	CRm	Opc2	Width	Description
CLUSTERPMOVSSET_EL1	3	c15	0	c5	3	32-bit	Cluster Overflow Flag Status Set
CLUSTERPMOVSCLR_EL1	3	c15	0	c5	4	32-bit	Cluster Overflow Flag Status Clear
CLUSTERPMSELR_EL1	3	c15	0	c5	5	32-bit	Cluster Event Counter Selection Register
CLUSTERPMINTENSET_EL1	3	c15	0	c5	6	32-bit	Cluster Interrupt Enable Set Register
CLUSTERPMINTENCLR_EL1	3	c15	0	c5	7	32-bit	Cluster Interrupt Enable Clear Register
CLUSTERPMXEVTYPER_EL1	3	c15	0	c6	1	32-bit	Cluster Selected Event Type and Filter Register
CLUSTERPMXEVCNTR_EL1	3	c15	0	c6	2	32-bit	Cluster Selected Event Counter Register
Reserved/RAZ	3	c15	0	c6	3	32-bit	Cluster Monitor Debug Configuration Register
CLUSTERPMCEID0_EL1	3	c15	0	c6	4	32-bit	Cluster Common Event Identification ID0 Register
CLUSTERPMCEID1_EL1	3	c15	0	c6	5	32-bit	Cluster Common Event Identification ID1 Register
CLUSTERPMCLAIMSET_EL1	3	c15	0	c6	6	32-bit	Cluster Performance Monitor Claim Tag Set Register
CLUSTERPMCLAIMCLR_EL1	3	c15	0	c6	7	32-bit	Cluster Performance Monitor Claim Tag Clear Register

Security

Name	Type	Description
ACTLR_EL3	RW	*B2.7 ACTLR_EL3, Auxiliary Control Register, EL3* on page B2-155
AFSR0_EL3	RW	*B2.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3* on page B2-159
AFSR1_EL3	RW	*B2.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3* on page B2-162
AMAIR_EL3	RW	*B2.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3* on page B2-166
CPTR_EL3	RW	*B2.37 CPTR_EL3, Architectural Feature Trap Register, EL3* on page B2-192
MDCR_EL3	RW	*B2.106 MDCR_EL3, Monitor Debug Configuration Register, EL3* on page B2-310

Reset management registers

Name	Type	Description
RMR_EL3	RW	*B2.111 RMR_EL3, Reset Management Register* on page B2-318
RVBAR_EL3	RW	*B2.112 RVBAR_EL3, Reset Vector Base Address Register, EL3* on page B2-319

Address registers

Name	Type	Description
PAR_EL1	RW	*B2.109 PAR_EL1, Physical Address Register, EL1* on page B2-316

B2.5 ACTLR_EL1, Auxiliary Control Register, EL1

ACTLR_EL1 provides IMPLEMENTATION DEFINED configuration and control options for execution at EL1 and EL0.

Bit field descriptions

ACTLR_EL1 is a 64-bit register, and is part of:

- The Other system control registers functional group
- The IMPLEMENTATION DEFINED functional group

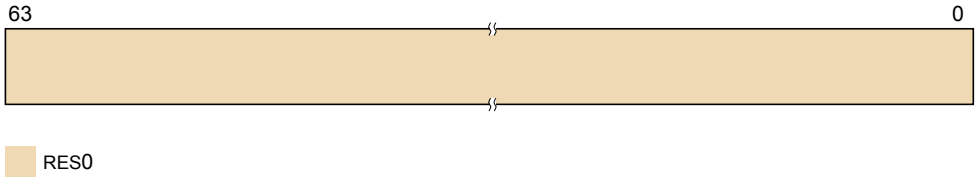

Figure B2-1 ACTLR_EL1 bit assignments

RES0, [63:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.6 ACTLR_EL2, Auxiliary Control Register, EL2

The ACTLR_EL2 provides IMPLEMENTATION DEFINED configuration and control options for EL2.

Bit field descriptions

ACTLR_EL2 is a 64-bit register, and is part of:

- The Virtualization registers functional group
- The Other system control registers functional group
- The IMPLEMENTATION DEFINED functional group

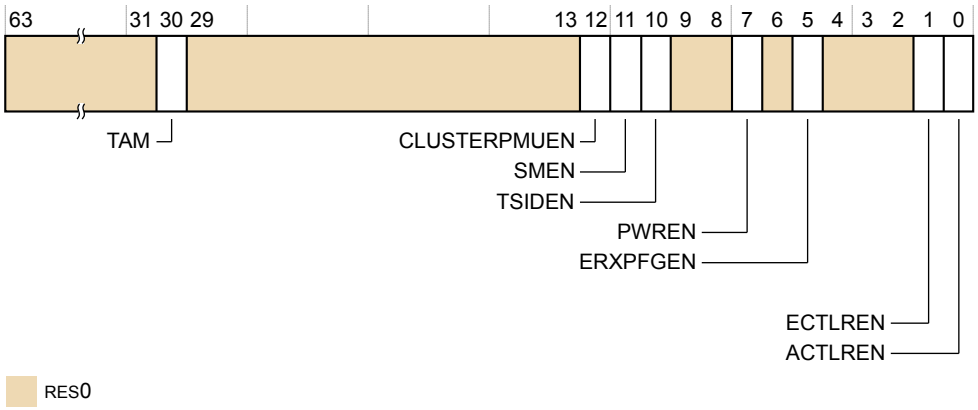

Figure B2-2 ACTLR_EL2 bit assignments

RES0, [63:31]

RES0 Reserved

TAM, [30]

Trap Activity Monitor registers. The possible values are:

0 El1 and EL0 accesses to all activity monitor registers are not trapped to EL2. This is the reset value.

1 EL1 and EL0 accesses to all activity monitor registers are trapped to EL2.

RES0, [29:13]

RES0 Reserved

CLUSTERPMUEN, [12]

Performance Management Registers enable. The possible values are:

0 CLUSTERPM* registers are not write-accessible from a lower Exception level. This is the reset value.

1 CLUSTERPM* registers are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

SMEN, [11]

Scheme Management Registers enable. The possible values are:

0 Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible from EL1 Non-secure. This is the reset value.

1 Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

TSIDEN, [10]

Thread Scheme ID Register enable. The possible values are:

0 Register CLUSTERTHREADSID is not write-accessible from EL1 Non-secure. This is the reset value.

1 Register CLUSTERTHREADSID is write-accessible from EL1 Non-secure if they are write-accessible from EL2.

RES0, [9:8]

RES0 Reserved

PWREN, [7]

Power Control Registers enable. The possible values are:

0 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write-accessible from EL1 Non-secure. This is the reset value.

1 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

RES0, [6]

RES0 Reserved

ERXPFGEN, [5]

Error Record Registers enable. The possible values are:

0 ERXPFG* are not write-accessible from EL1 Non-secure. This is the reset value.

1 ERXPFG* are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

RES0, [4:2]

RES0 Reserved

ECTLREN, [1]

Extended Control Registers enable. The possible values are:

0 CPUECTLR* and CLUSTERECTLR are not write-accessible from EL1 Non-secure. This is the reset value.

1 CPUECTLR* and CLUSTERECTLR are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

ACTLREN, [0]

Auxiliary Control Registers enable. The possible values are:

0 CPUACTLR*, CPUACTLR2, CPUACTLR3, and CLUSTERACTLR are not write-accessible from EL1 Non-secure. This is the reset value.

1 CPUACTLR*, CPUACTLR2, CPUACTLR3, and CLUSTERACTLR are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

Configurations

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.7 ACTLR_EL3, Auxiliary Control Register, EL3

The ACTLR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for EL3.

Bit field descriptions

ACTLR_EL3 is a 64-bit register, and is part of:

- The Other system control registers functional group
- The Security registers functional group
- The IMPLEMENTATION DEFINED functional group

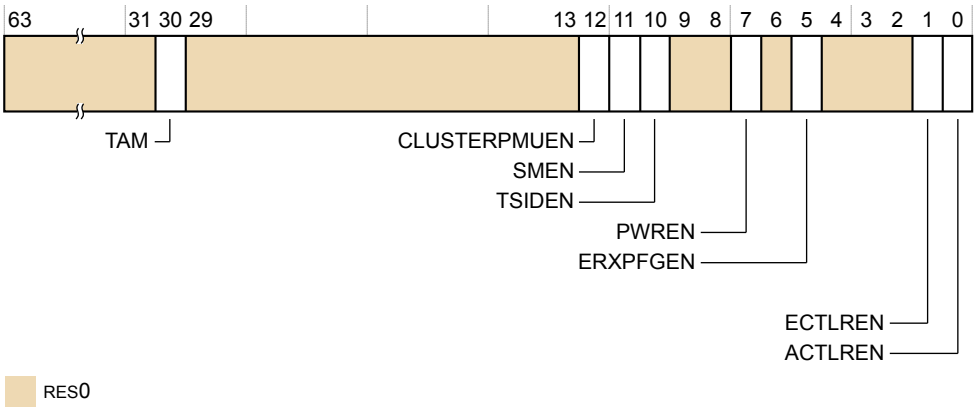

Figure B2-3 ACTLR_EL3 bit assignments

RES0, [63:31]

RES0 Reserved

TAM, [30]

Trap Activity Monitor registers. The possible values are:

0 EL2, EL1, and EL0 accesses to all activity monitor registers are not trapped to EL3. This is the reset value.

1 EL2, EL1, and EL0 accesses to all activity monitor registers are trapped to EL3.

RES0, [29:13]

RES0 Reserved

CLUSTERPMUEN, [12]

Performance Management Registers enable. The possible values are:

0 CLUSTERPM* registers are not write-accessible from a lower Exception level. This is the reset value.

1 CLUSTERPM* registers are write-accessible from EL2 and EL1 Secure.

SMEN, [11]

Scheme Management Registers enable. The possible values are:

0 Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible from EL2 and EL1 Secure.

TSIDEN, [10]

Thread Scheme ID Register enable. The possible values are:

0 Register CLUSTERTHREADSID is not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 Register CLUSTERTHREADSID is write-accessible from EL2 and EL1 Secure.

RES0, [9:8]

RES0 Reserved

PWREN, [7]

Power Control Registers enable. The possible values are:

0 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL2 and EL1 Secure.

RES0, [6]

RES0 Reserved

ERXPFGEN, [5]

Error Record Registers enable. The possible values are:

0 ERXPFG* are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 ERXPFG* are write-accessible from EL2 and EL1 Secure.

RES0, [4:2]

RES0 Reserved

ECTLREN, [1]

Extended Control Registers enable. The possible values are:

0 CPUECTLR and CLUSTERECTLR are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 CPUECTLR and CLUSTERECTLR are write-accessible from EL2 and EL1 Secure.

ACTLREN, [0]

Auxiliary Control Registers enable. The possible values are:

0 CPUACTLR*, CPUACTLR2, CPUACTLR3, and CLUSTERACTLR are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 CPUACTLR*, CPUACTLR2, CPUACTLR3, and CLUSTERACTLR are write-accessible from EL2 and EL1 Secure.

Configurations

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1

AFSR0_EL1 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL1. In the Cortex-A78C core, no additional information is provided for these exceptions. Therefore this register is not used.

Bit field descriptions

AFSR0_EL1 is a 32-bit register, and is part of:

- The Exception and fault handling registers functional group
- The IMPLEMENTATION DEFINED functional group

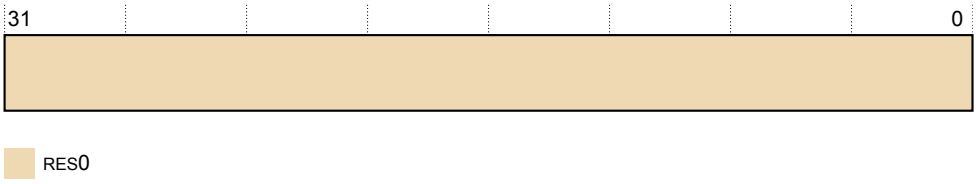

Figure B2-4 AFSR0_EL1 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2

AFSR0_EL2 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL2.

Bit field descriptions

AFSR0_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group
- The Exception and fault handling registers functional group
- The IMPLEMENTATION DEFINED functional group

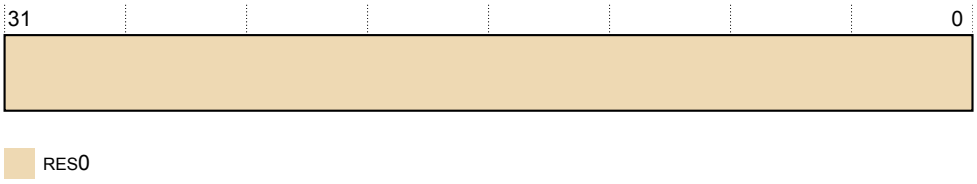

Figure B2-5 AFSR0_EL2 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3

AFSR0_EL3 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL3. In the Cortex-A78C core, no additional information is provided for these exceptions. Therefore this register is not used.

Bit field descriptions

AFSR0_EL3 is a 32-bit register, and is part of:

- The Exception and fault handling registers functional group
- The Security registers functional group
- The IMPLEMENTATION DEFINED functional group

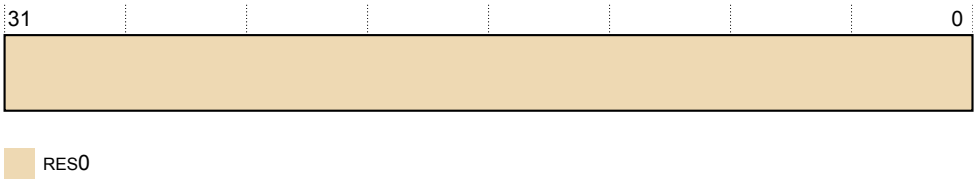

Figure B2-6 AFSR0_EL3 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1

AFSR1_EL1 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL1. This register is not used in Cortex-A78C.

Bit field descriptions

AFSR1_EL1 is a 32-bit register, and is part of:

- The Exception and fault handling registers functional group
- The IMPLEMENTATION DEFINED functional group

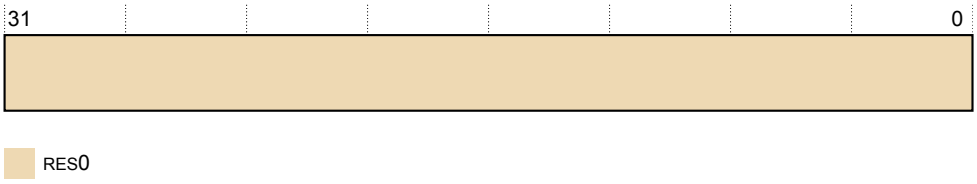

Figure B2-7 AFSR1_EL1 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2

AFSR1_EL2 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL2. This register is not used in the Cortex-A78C core.

Bit field descriptions

AFSR1_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group
- The Exception and fault handling registers functional group
- The IMPLEMENTATION DEFINED functional group

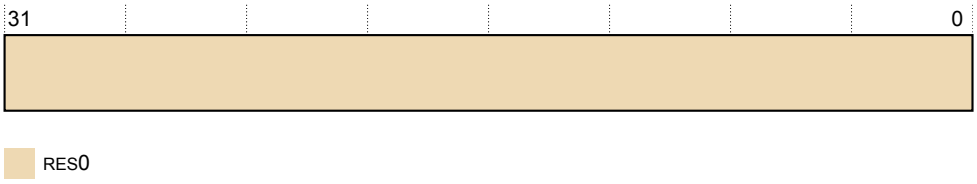

Figure B2-8 AFSR1_EL2 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3

AFSR1_EL3 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL3. This register is not used in the Cortex-A78C core.

Bit field descriptions

AFSR1_EL3 is a 32-bit register, and is part of:

- The Exception and fault handling registers functional group
- The Security registers functional group
- The IMPLEMENTATION DEFINED functional group

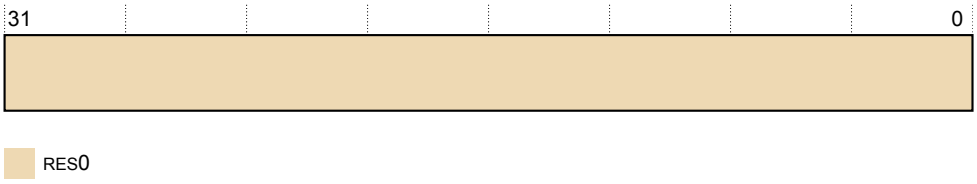

Figure B2-9 AFSR1_EL3 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.14 AIDR_EL1, Auxiliary ID Register, EL1

AIDR_EL1 provides IMPLEMENTATION DEFINED identification information. This register is not used in the Cortex-A78C core.

Bit field descriptions

AIDR_EL1 is a 32-bit register, and is part of:

- The Identification registers functional group
- The IMPLEMENTATION DEFINED functional group

This register is read-only.

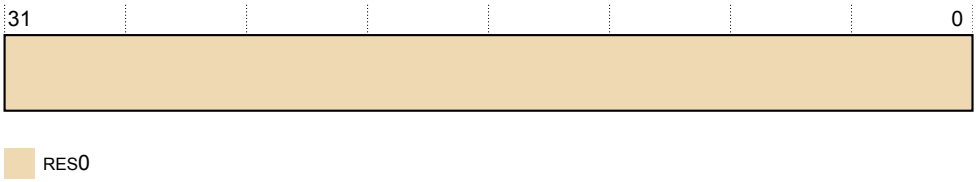

Figure B2-10 AIDR_EL1 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1

AMAIR_EL1 provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by MAIR_EL1. This register is not used in the Cortex-A78C core.

Bit field descriptions

AMAIR_EL1 is a 64-bit register, and is part of:

- The Virtual memory control registers functional group
- The IMPLEMENTATION DEFINED functional group

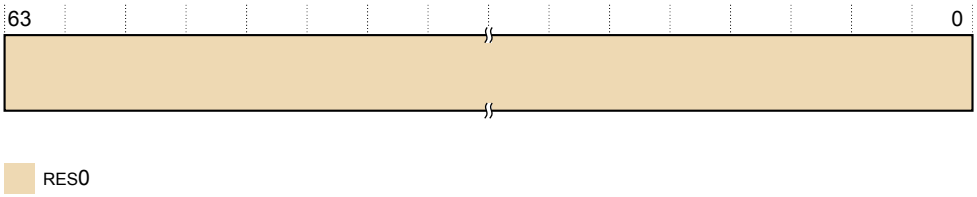

Figure B2-11 AMAIR_EL1 bit assignments

RES0, [63:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2

AMAIR_EL2 provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by MAIR_EL2. This register is not used in the Cortex-A78C core.

Bit field descriptions

AMAIR_EL2 is a 64-bit register, and is part of:

- The Virtualization registers functional group
- The Virtual memory control registers functional group
- The IMPLEMENTATION DEFINED functional group

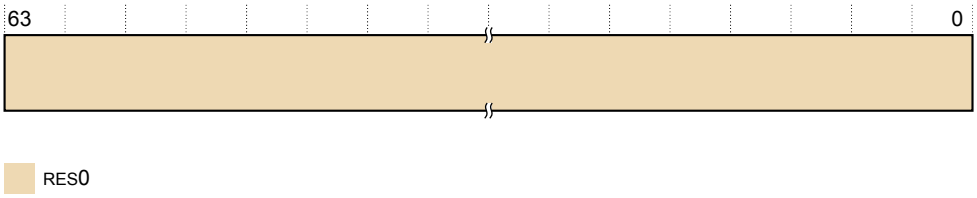

Figure B2-12 AMAIR_EL1 bit assignments

RES0, [63:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3

AMAIR_EL3 provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by MAIR_EL3. This register is not used in the Cortex-A78C core.

Bit field descriptions

AMAIR_EL3 is a 64-bit register, and is part of:

- The Virtual memory control registers functional group
- The Security registers functional group
- The IMPLEMENTATION DEFINED functional group

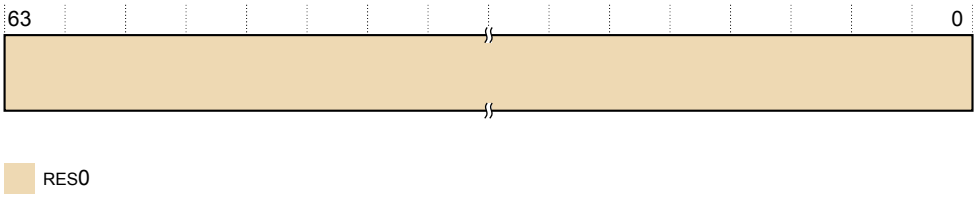

Figure B2-13 AMAIR_EL3 bit assignments

RES0, [63:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.18 APDAKeyHi_EL1, Pointer Authentication Key A for Data

APDAKeyHi_EL1 holds bits [63:0] of key A used for authentication of instruction pointer values.

Bit field descriptions

The APDAKeyHi_EL1 is a 64-bit register.

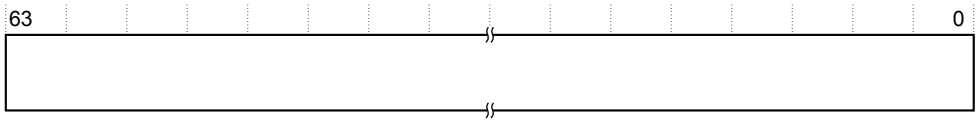

Figure B2-14 APDAKeyHi_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APDAKeyHi_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APDAKeyHi_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-6 APDAKeyHi_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C2_1	11	000	0010	0010	001

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C2_1	-	RW	n/a	RW
S3_0_C2_C2_1	-	RW	RW	RW
S3_0_C2_C2_1	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.19 APDAKeyLo_EL1, Pointer Authentication Key A for Data

APDAKeyLo_EL1 holds bits [63:0] of key A used for authentication of instruction pointer values.

Bit field descriptions

The APDAKeyLo_EL1 is a 64-bit register.

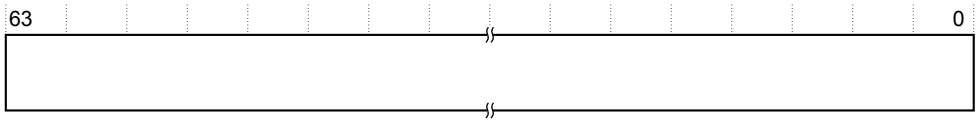

Figure B2-15 APDAKeyLo_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APDAKeyLo_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APDAKeyLo_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-7 APDAKeyLo_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C2_0	11	000	0010	0010	000

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C2_0	-	RW	n/a	RW
S3_0_C2_C2_0	-	RW	RW	RW
S3_0_C2_C2_0	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.20 APDBKeyHi_EL1, Pointer Authentication Key B for Data

APDBKeyHi_EL1 holds bits [63:0] of key B used for authentication of instruction pointer values.

Bit field descriptions

The APDBKeyHi_EL1 is a 64-bit register.

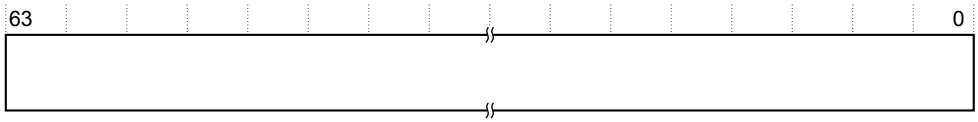

Figure B2-16 APDBKeyHi_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APDBKeyHi_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APDBKeyHi_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-8 APDBKeyHi_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C2_3	11	000	0010	0010	011

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C2_3	-	RW	n/a	RW
S3_0_C2_C2_3	-	RW	RW	RW
S3_0_C2_C2_3	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.21 APDBKeyLo_EL1, Pointer Authentication Key B for Data

APDBKeyLo_EL1 holds bits [63:0] of key B used for authentication of instruction pointer values.

Bit field descriptions

The APDBKeyLo_EL1 is a 64-bit register.

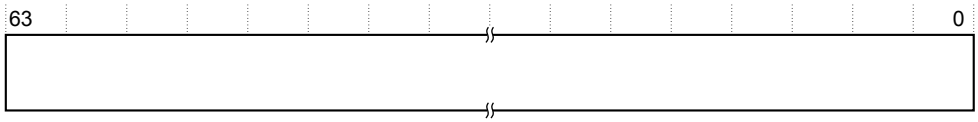

Figure B2-17 APDBKeyLo_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APDBKeyLo_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APDBKeyLo_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-9 APDBKeyLo_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C2_2	11	000	0010	0010	010

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C2_2	-	RW	n/a	RW
S3_0_C2_C2_2	-	RW	RW	RW
S3_0_C2_C2_2	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.22 APGAKeyHi_EL1, Pointer Authentication Key A for Code

APGAKeyHi_EL1 holds bits [63:0] of key A used for authentication of instruction pointer values.

Bit field descriptions

The APGAKeyHi_EL1 is a 64-bit register.

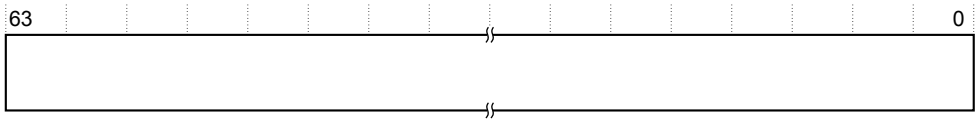

Figure B2-18 APGAKeyHi_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APGAKeyHi_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APGAKeyHi_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-10 APGAKeyHi_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C3_1	11	000	0010	0011	001

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C3_1	-	RW	n/a	RW
S3_0_C2_C3_1	-	RW	RW	RW
S3_0_C2_C3_1	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.23 APGAKeyLo_EL1, Pointer Authentication Key A for Code

APGAKeyLo_EL1 holds bits [63:0] of key A used for authentication of instruction pointer values.

Bit field descriptions

The APGAKeyLo_EL1 is a 64-bit register.

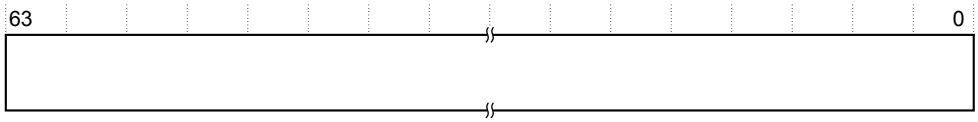

Figure B2-19 APGAKeyLo_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APGAKeyLo_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APGAKeyLo_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-11 APGAKeyLo_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C3_0	11	000	0010	0011	000

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C3_0	-	RW	n/a	RW
S3_0_C2_C3_0	-	RW	RW	RW
S3_0_C2_C3_0	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.24 APIAKeyHi_EL1, Pointer Authentication Key A for Instruction

APIAKeyHi_EL1 holds bits [63:0] of key A used for authentication of instruction pointer values.

Bit field descriptions

The APIAKeyHi_EL1 is a 64-bit register.

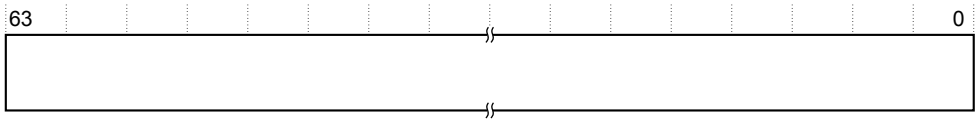

Figure B2-20 APIAKeyHi_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APIAKeyHi_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APIAKeyHi_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MSR <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-12 APIAKeyHi_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C1_1	11	000	0010	0001	001

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C1_1	-	RW	n/a	RW
S3_0_C2_C1_1	-	RW	RW	RW
S3_0_C2_C1_1	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.25 APIAKeyLo_EL1, Pointer Authentication Key A for Instruction

APIAKeyLo_EL1 holds bits [63:0] of key A used for authentication of instruction pointer values.

Bit field descriptions

The APIAKeyLo_EL1 is a 64-bit register.

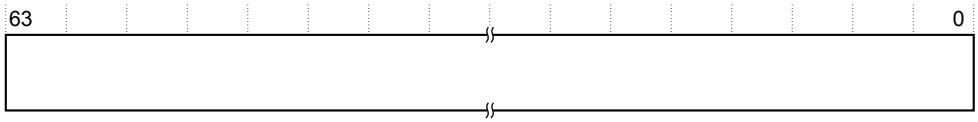

Figure B2-21 APIAKeyLo_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APIAKeyLo_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APIAKeyLo_EL1

This register can be written using MSR (register) with the following syntax:

```
MRS <systemreg>, <Xt>
```

This register can be written using MSR (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-13 APIAKeyLo_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C1_0	11	000	0010	0001	000

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C1_0	-	RW	n/a	RW
S3_0_C2_C1_0	-	RW	RW	RW
S3_0_C2_C1_0	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.26 APIBKeyHi_EL1, Pointer Authentication Key B for Instruction

APIBKeyHi_EL1 holds bits [63:0] of key B used for authentication of instruction pointer values.

Bit field descriptions

The APIBKeyHi_EL1 is a 64-bit register.

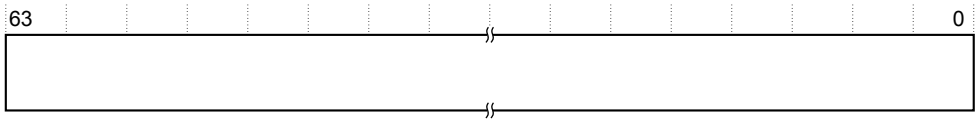

Figure B2-22 APIBKeyHi_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APIBKeyHi_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APIBKeyHi_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-14 APIBKeyHi_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C1_3	11	000	0010	0001	011

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C1_3	-	RW	n/a	RW
S3_0_C2_C1_3	-	RW	RW	RW
S3_0_C2_C1_3	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.27 APIBKeyLo_EL1, Pointer Authentication Key B for Instruction

APIBKeyLo_EL1 holds bits [63:0] of key B used for authentication of instruction pointer values.

Bit field descriptions

The APIBKeyLo_EL1 is a 64-bit register.

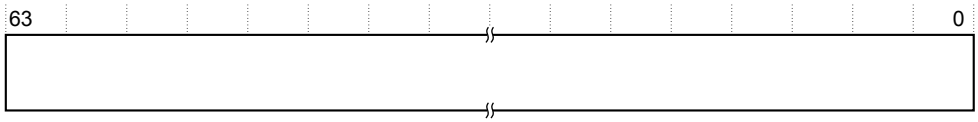

Figure B2-23 APIBKeyLo_EL1 bit assignments

Bits [63:0]

64 bit value, bits[63:0] of the 128 bit pointer authentication key value.

This field resets to an architecturally UNKNOWN value.

Configurations

This register is present only when ARMv8.3-PAuth is implemented. Otherwise, direct accesses to APIBKeyLo_EL1 are UNDEFINED. RW fields in this register reset to architecturally UNKNOWN values.

Usage constraints

Accessing APIBKeyLo_EL1

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This register can be written using MRS (register) with the following syntax:

```
MRS <Xt>, <systemreg>
```

Register access is encoded as follows:

Table B2-15 APIBKeyLo_EL1 encoding

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C2_C1_2	11	000	0010	0001	010

This register is accessible as follows:

<systemreg>	EL0	EL1	EL2	EL3
S3_0_C2_C1_2	-	RW	n/a	RW
S3_0_C2_C1_2	-	RW	RW	RW
S3_0_C2_C1_2	-	n/a	RW	RW

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

B2.28 ATCR_EL1, Auxiliary Translation Control Register, EL1

The ATCR_EL1 determines the values of PBHA on page table walks memory access in EL1 translation regime.

Bit field descriptions

ATCR_EL1 is a 64-bit register.

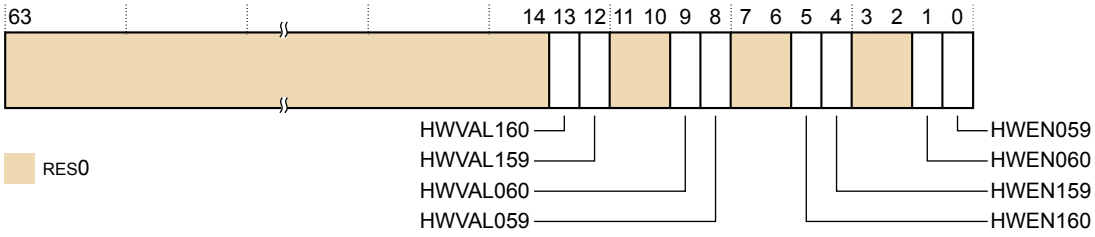

Figure B2-24 ATCR_EL1 bit assignments

RES0, [63:14]

RES0 Reserved

HWVAL160, [13]

Indicates the value of PBHA[1] on page table walks memory access targeting the base address defined by TTBR1_EL1 if HWEN160 is set.

HWVAL159, [12]

Indicates the value of PBHA[0] on page table walks memory access targeting the base address defined by TTBR1_EL1 if HWEN159 is set.

RES0, [11:10]

RES0 Reserved

HWVAL060, [9]

Indicates the value of PBHA[1] page table walks memory access targeting the base address defined by TTBR0_EL1 if HWEN060 is set.

HWVAL059, [8]

Indicates the value of PBHA[1] page table walks memory access targeting the base address defined by TTBR0_EL1 if HWEN059 is set.

RES0, [7:6]

RES0 Reserved

HWEN160, [5]

Enables PBHA[1] page table walks memory access targeting the base address defined by TTBR1_EL1. If this bit is clear, PBHA[1] on page table walks is 0.

HWEN159, [4]

Enables PBHA[0] page table walks memory access targeting the base address defined by TTBR1_EL1. If this bit is clear, PBHA[0] on page table walks is 0.

RES0, [3:2]

RES0 Reserved

HWEN060, [1]

Enables PBHA[1] page table walks memory access targeting the base address defined by TTBR0_EL1. If this bit is clear, PBHA[1] on page table walks is 0.

HWEN059, [0]

Enables PBHA[0] page table walks memory access targeting the base address defined by TTBR0_EL1. If this bit is clear, PBHA[0] on page table walks is 0.

Configurations

AArch64 register ATCR_EL1 is mapped to AArch32 register ATTBCR (NS).

At EL2 with HCR_EL2.E2H set, accesses to ATCR_EL1 are remapped to access ATCR_EL2.

Usage constraints

Accessing the ATCR_EL1

To access the ATCR_EL1:

```
MRS Xt , S< 3   0  c15   c7  0> ; Read ATCR_EL1 into Xt
MSR S < 3   0  c15   c7  0 > , Xt   ; Write Xt to ATCR_EL1
```

This syntax is encoded with the following settings in the instruction encoding:

Op0	Op1	CRn	CRm	Op2
3	0	c15	c7	0

Accessibility

ATCR_EL1 is accessible as follows:

	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
ATCR_EL1	x	x	0	-	RW	n/a	RW
ATCR_EL1	0	0	1	-	RW	RW	RW
ATCR_EL1	0	1	1	-	n/a	RW	RW
ATCR_EL1	1	0	1	-	RW	ATCR_EL2	RW
ATCR_EL1	1	1	1	-	n/a	ATCR_EL2	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Note

ATCR_EL1 is also accessible using ATCR_EL12 when HCR.EL2.E2H is set. See *B2.30 ATCR_EL12 , Alias to Auxiliary Translation Control Register EL1* on page B2-181.

Traps and enables

Rules of traps and enables for this register are the same as TCR_EL1. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.29 ATCR_EL2, Auxiliary Translation Control Register, EL2

The ATCR_EL2 determines the values of PBHA on page table walks memory access in EL2 translation regime.

Bit field descriptions

ATCR_EL2 is a 64-bit register.

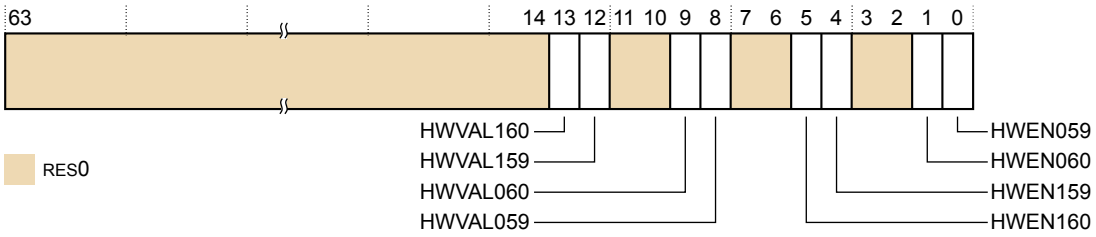

Figure B2-25 ATCR_EL2 bit assignments

RES0, [63:14]

RES0 Reserved

HWVAL160, [13]

Indicates the value of PBHA[1] on page table walks memory access targeting the base address defined by TTBR1_EL2 if HWEN160 is set.

HWVAL159, [12]

Indicates the value of PBHA[0] on page table walks memory access targeting the base address defined by TTBR1_EL2 if HWEN159 is set.

RES0, [11:10]

RES0 Reserved

HWVAL060, [9]

Indicates the value of PBHA[1] page table walks memory access targeting the base address defined by TTBR0_EL2 if HWEN060 is set.

HWVAL059, [8]

Indicates the value of PBHA[1] page table walks memory access targeting the base address defined by TTBR0_EL2 if HWEN059 is set.

RES0, [7:6]

RES0 Reserved

HWEN160, [5]

Enables PBHA[1] page table walks memory access targeting the base address defined by TTBR1_EL2. If this bit is clear, PBHA[1] on page table walks is 0.

HWEN159, [4]

Enables PBHA[0] page table walks memory access targeting the base address defined by TTBR1_EL2. If this bit is clear, PBHA[0] on page table walks is 0.

RES0, [3:2]

RES0 Reserved

HWEN060, [1]

Enables PBHA[1] page table walks memory access targeting the base address defined by TTBR0_EL2. If this bit is clear, PBHA[1] on page table walks is 0.

HWEN059, [0]

Enables PBHA[0] page table walks memory access targeting the base address defined by TTBR0_EL2. If this bit is clear, PBHA[0] on page table walks is 0.

Configurations

AArch64 ATCR_EL2 register is architecturally mapped to AArch32 register AHTCR.

Usage constraints

Accessing the ATCR_EL2

To access the ATCR_EL2:

```
MRS Xt, S< 3   4  c15   c7  0> ; Read ATCR_EL2 into Xt
MSR S < 3   4 c15   c7 0 > , Xt   ; Write Xt to ATCR_EL2
```

This syntax is encoded with the following settings in the instruction encoding:

Op0	Op1	CRn	CRm	Op2
3	4	c15	c7	0

Accessibility

ATCR_EL2 is accessible as follows:

EL0 (NS)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS=1)	EL3 (SCR.NS=0)
-	-	-	RW	RW	RW

B2.30 ATCR_EL12 , Alias to Auxiliary Translation Control Register EL1

The ATCR_EL12 alias allows access to ATCR_EL1 at EL2 or EL3 when HCR_EL2.E2H is set to 1.

Usage constraints

Accessing the ATCR_EL12

To access the ATCR_EL1using the ATCR_EL12 alias:

```
MRS Xt , S< 3   5  c15   c7  0> ; Read ATCR_EL12/ATCR_EL1 into Xt
                   MSR S < 3   5  c15   c7  0 > , Xt   ; Write Xt to
ATCR_EL12/ATCR_EL1
```

This syntax is encoded with the following settings in the instruction encoding:

Op0	Op1	CRn	CRm	Op2
3	5	15	7	0

Accessibility

ATCR_EL12 is accessible as follows:

	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
ATCR_EL12	x	x	0	-	-	n/a	-
ATCR_EL12	0	0	1	-	-	-	-
ATCR_EL12	0	1	1	-	n/a	-	-
ATCR_EL12	1	0	1	-	-	ATCR_EL1	ATCR_EL1
ATCR_EL12	1	1	1	-	n/a	ATCR_EL1	ATCR_EL1

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

All traps associated with the ATCR_EL1 register that apply at EL2 or EL3 also apply to the ATCR_EL12 alias.

This alias is only accessible when HCR_EL2.E2H == 1.

When HCR_EL2.E2H == 0, access to this alias is UNDEFINED.

B2.31 ATCR_EL3, Auxiliary Translation Control Register, EL3

The ATCR_EL3 determines the values of PBHA on page table walks memory access in EL3 translation regime.

Bit field descriptions

ATCR_EL3 is a 64-bit register.

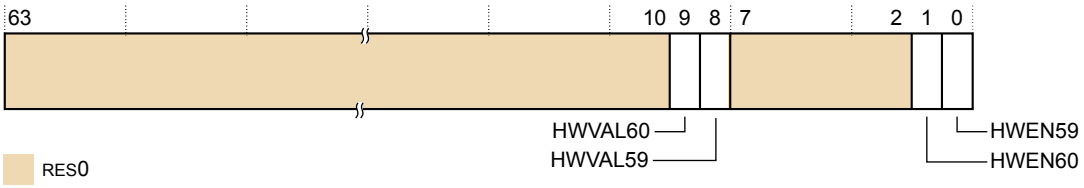

Figure B2-26 ATCR_EL3 bit assignments

RES0, [63:10]

RES0 Reserved

HWVAL60, [9]

Indicates the value of PBHA[1] page table walks memory access if HWEN60 is set.

HWVAL59, [8]

Indicates the value of PBHA[1] page table walks memory access if HWEN59 is set.

RES0, [7:2]

RES0 Reserved

HWEN60, [1]

Enables PBHA[1] page table walks memory access. If this bit is clear, PBHA[1] on page table walks is 0.

HWEN59, [0]

Enables PBHA[0] page table walks memory access. If this bit is clear, PBHA[0] on page table walks is 0.

Configurations

AArch64 register ATCR_EL3 is architecturally mapped to AArch32 register ATCR (S).

Usage constraints

Accessing the ATCR_EL3

To access the ATCR_EL3:

```
MRS Xt , < 3  6 c15  c7 0>  ; Read ATCR_EL3 into Xt
MSR S < 3   6 c15   c7 0 > , Xt   ; Write Xt to ATCR_EL3
```

This syntax is encoded with the following settings in the instruction encoding:

Op0	Op1	CRn	CRm	Op2
3	6	c15	c7	0

Accessibility

ATCR_EL3 is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS=1)	EL3 (SCR.NS=0)
-	-	-	-	RW	RW

B2.32 AVTCR_EL2, Auxiliary Virtualized Translation Control Register, EL2

The AVTCR_EL2 determines the values of PBHA on stage 2 page table walks memory access in EL1 Non-secure translation regime if stage 2 is enable.

Bit field descriptions

AVTCR_EL2 is a 64-bit register.

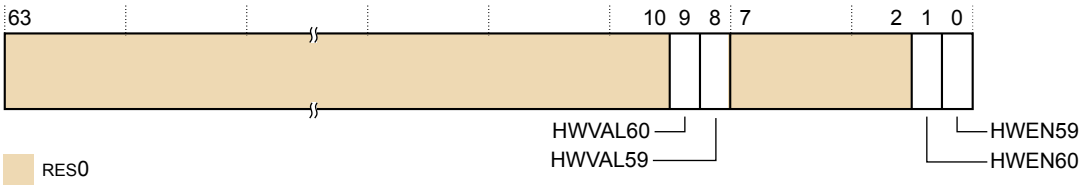

Figure B2-27 AVTCR_EL2 bit assignments

RES0, [63:10]

RES0 Reserved

HWVAL60, [9]

Indicates the value of PBHA[1] page table walks memory access if HWEN60 is set.

HWVAL59, [8]

Indicates the value of PBHA[1] page table walks memory access if HWEN59 is set.

RES0, [7:2]

RES0 Reserved

HWEN60, [1]

Enables PBHA[1] page table walks memory access. If this bit is clear, PBHA[1] on page table walks is 0.

HWEN59, [0]

Enables PBHA[0] page table walks memory access. If this bit is clear, PBHA[0] on page table walks is 0.

Configurations

AArch64 register AVTCR_EL2 is architecturally mapped to AArch32 register AVTCR.

Usage constraints

Accessing the AVTCR_EL2

To access the AVTCR_EL2:

```
MRS  Xt , S< 3  4  c15  c7  1> ; Read AVTCR_EL2 into Xt
MSR S < 3   4  c15   c7  1 > , Xt   ; Write Xt to AVTCR_EL2
```

This syntax is encoded with the following settings in the instruction encoding:

Op0	Op1	CRn	CRm	Op2
3	4	c15	c7	1

Accessibility

AVTCR_EL2 is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS=1)	EL3 (SCR.NS=0)
-	-	-	RW	RW	RW

B2.33 CCSIDR_EL1, Cache Size ID Register, EL1

The CCSIDR_EL1 provides information about the architecture of the currently selected cache.

Bit field descriptions

CCSIDR_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

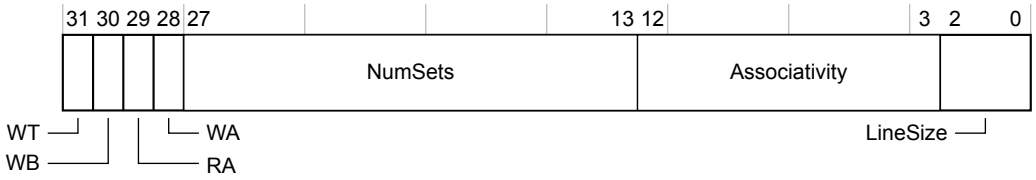

Figure B2-28 CCSIDR_EL1 bit assignments

WT, [31]

Indicates whether the selected cache level supports Write-Through:

`0` Cache Write-Through is not supported at any level.

For more information about encoding, see *CCSIDR_EL1 encodings* on page B2-187.

WB, [30]

Indicates whether the selected cache level supports Write-Back. Permitted values are:

`0` Write-Back is not supported.

`1` Write-Back is supported.

For more information about encoding, see *CCSIDR_EL1 encodings* on page B2-187.

RA, [29]

Indicates whether the selected cache level supports read-allocation. Permitted values are:

`0` Read-allocation is not supported.

`1` Read-allocation is supported.

For more information about encoding, see *CCSIDR_EL1 encodings* on page B2-187.

WA, [28]

Indicates whether the selected cache level supports write-allocation. Permitted values are:

`0` Write-allocation is not supported.

`1` Write-allocation is supported.

For more information about encoding, see *CCSIDR_EL1 encodings* on page B2-187.

NumSets, [27:13]

(Number of sets in cache) - 1. Therefore, a value of 0 indicates one set in the cache. The number of sets does not have to be a power of 2.

For more information about encoding, see *CCSIDR_EL1 encodings* on page B2-187.

Associativity, [12:3]

(Associativity of cache) - 1. Therefore, a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.

For more information about encoding, see *CCSIDR_EL1 encodings* on page B2-187.

LineSize, [2:0]

(Log_2(Number of bytes in cache line)) - 4. For example:

For a line length of 16 bytes: $Log_2(16) = 4$, LineSize entry = 0. This is the minimum line length.

For a line length of 32 bytes: $Log_2(32) = 5$, LineSize entry = 1.

For more information about encoding, see *CCSIDR_EL1 encodings* on page B2-187.

Configurations

There are no configuration notes.
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

CCSIDR_EL1 encodings

The following table shows the individual bit field and complete register encodings for the CCSIDR_EL1.

Table B2-16 CCSIDR encodings

CSSELR		Cache	Size	Complete register encoding	Register bit field encoding						
Level	InD				WT	WB	RA	WA	NumSets	Associativity	LineSize
0b000	0b0	L1 Data cache	64KB	701FE01A	0	1	1	1	0x00FF	0x003	2
0b000	0b1	L1 Instruction cache	64KB	201FE01A	0	0	1	0	0x00FF	0x003	2
0b001	0b0	L2 cache	256KB	703FE03A	0	1	1	1	0x01FF	0x007	2
			512KB	707FE03A	0	1	1	1	0x03FF	0x007	2
0b001	0b1	Reserved	-	-	-	-	-	-	-	-	-
0b010	0b0	L3 cache	256KB	701FE07A	0	1	1	1	000F	00F	2
			512KB	703FE07A					01FF	00F	2
			1MB	707FE07A					03FF	00F	2
			2MB	70FFE07A					07FF	00F	2
			4MB	71FFE07A					0FFF	00F	2
			8MB	73FFE07A					1FFF	00F	2
0b0101 - 0b1111		Reserved	-	-	-	-	-	-	-	-	-

B2.34 CLIDR_EL1, Cache Level ID Register, EL1

The CLIDR_EL1 identifies the type of cache, or caches, implemented at each level, up to a maximum of seven levels.

It also identifies the *Level of Coherency* (LoC) and *Level of Unification* (LoU) for the cache hierarchy.

Bit field descriptions

CLIDR_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

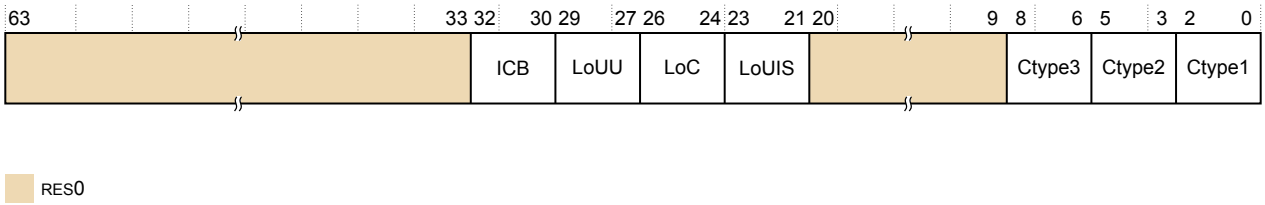

Figure B2-29 CLIDR_EL1 bit assignments

RES0, [63:33]

RES0 Reserved

ICB, [32:30]

Inner cache boundary. This field indicates the boundary between the inner and the outer domain:

`0b010` L2 cache is the highest inner level.

`0b011` L3 cache is the highest inner level.

LoUU, [29:27]

Indicates the Level of Unification Uniprocessor for the cache hierarchy:

`0b000` No levels of cache need to cleaned or invalidated when cleaning or invalidating to the Point of Unification. This is the value if no caches are configured.

LoC, [26:24]

Indicates the Level of Coherency for the cache hierarchy:

`0b010` L3 cache is not implemented.

`0b011` L3 cache is implemented.

LoUIS, [23:21]

Indicates the *Level of Unification Inner Shareable* (LoUIS) for the cache hierarchy.

`0b000` No cache level needs cleaning to Point of Unification.

RES0, [20:9]

No cache at levels L7 down to L4.

RES0 Reserved

Ctype3, [8:6]

Indicates the type of cache if the core implements L3 cache. If present, unified instruction and data caches at L3:

`0b100` Both per-core L2 and cluster L3 caches are present.

`0b000` All other options

If Ctype2 has a value of `0b000`, then the value of Ctype3 must be IGNORED.

Ctype2, [5:3]

Indicates the type of unified instruction and data caches at L2:

`0b100` Either per-core L2 or cluster L2 cache is present.

`0b000` All other options

Ctype1, [2:0]

Indicates the type of cache implemented at L1:

`0b011` Separate instruction and data caches at L1

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.35 CPACR_EL1, Architectural Feature Access Control Register, EL1

The CPACR_EL1 controls access to trace functionality and access to registers associated with Advanced SIMD and floating-point execution.

Bit field descriptions

CPACR_EL1 is a 32-bit register, and is part of the Other system control registers functional group.

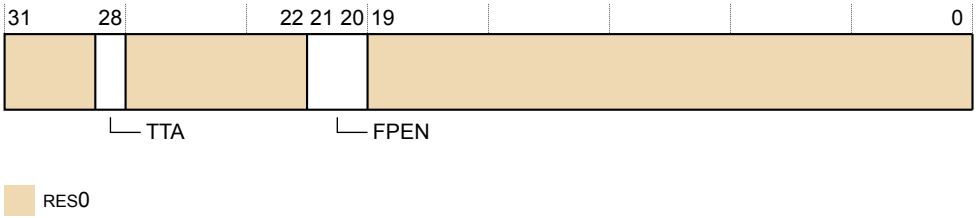

Figure B2-30 CPACR_EL1 bit assignments

RES0, [31:29]

RES0 Reserved

TTA, [28]

Traps EL0 and EL1 System register accesses to all implemented trace registers to EL1, from both Execution states. This bit is RES0. The core does not provide System Register access to ETM control.

Configurations

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.36 CPTR_EL2, Architectural Feature Trap Register, EL2

The CPTR_EL2 controls trapping to EL2 for accesses to CPACR, trace functionality and registers associated with Advanced SIMD and floating-point execution. It also controls EL2 access to this functionality.

Bit field descriptions

CPTR_EL2 is a 32-bit register, and is part of the Virtualization registers functional group.

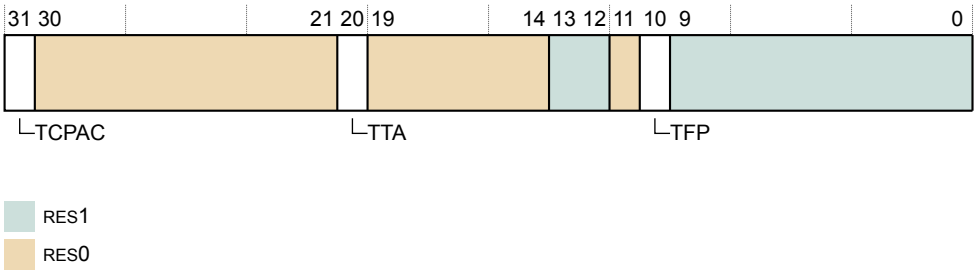

Figure B2-31 CPTR_EL2 bit assignments

TTA, [20]

Trap Trace Access

This bit is not implemented. RES0.

Configurations

RW fields in this register reset to UNKNOWN values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.37 CPTR_EL3, Architectural Feature Trap Register, EL3

The CPTR_EL3 controls trapping to EL3 of access to CPACR_EL1, CPTR_EL2, trace functionality and registers associated with Advanced SIMD and floating-point execution.

It also controls EL3 access to trace functionality and registers associated with Advanced SIMD and floating-point execution.

Bit field descriptions

CPTR_EL3 is a 32-bit register, and is part of the Security registers functional group.

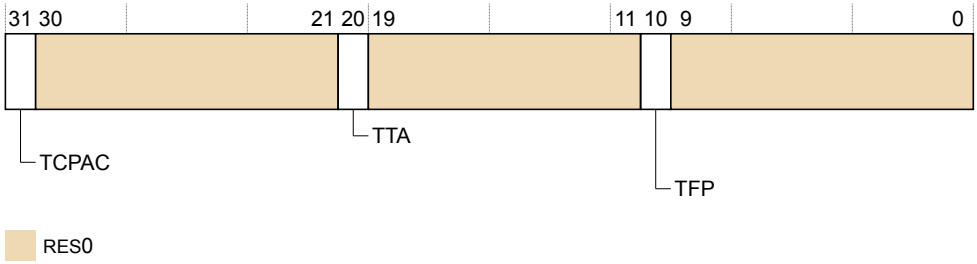

Figure B2-32 CPTR_EL3 bit assignments

TTA, [20]

Trap Trace Access

Not implemented. RES0.

TFP, [10]

Traps all accesses to SVE, Advanced SIMD and floating-point functionality to EL3. This applies to all Exception levels, both Security states, and both Execution states. The possible values are:

`0` Does not cause any instruction to be trapped. This is the reset value.

`1` Any attempt at any Exception level to execute an instruction that uses the registers that are associated with SVE, Advanced SIMD and floating-point is trapped to EL3, subject to the exception prioritization rules.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.38 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1

The CPUACTLR_EL1 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUACTLR_EL1 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

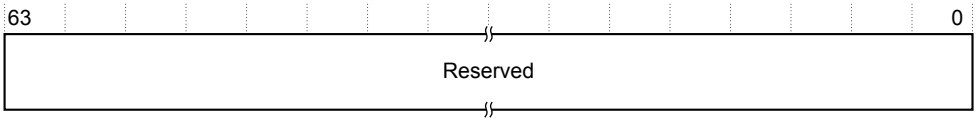

Figure B2-33 CPUACTLR_EL1 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUACTLR_EL1 is common to the Secure and Non-secure states.

Usage constraints

Accessing the CPUACTLR_EL1

The CPU Auxiliary Control Register can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Setting many of these bits can cause significantly lower performance on your code. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_0	11	000	1111	0001	000

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C1_0	x	x	0	-	RW	n/a	RW
S3_0_C15_C1_0	x	0	1	-	RW	RW	RW
S3_0_C15_C1_0	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.39 CPUACTLR2_EL1, CPU Auxiliary Control Register 2, EL1

The CPUACTLR2_EL1 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUACTLR2_EL1 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

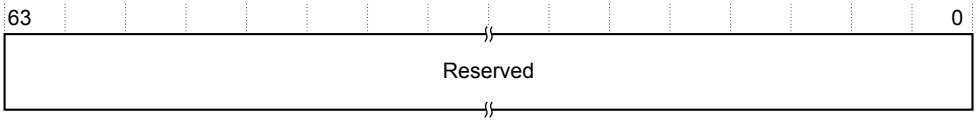

Figure B2-34 CPUACTLR2_EL1 bit assignments

Reserved, [63:0]
Reserved for Arm internal use.

Configurations

CPUACTLR2_EL1 is common to the Secure and Non-secure states.

Usage constraints

Accessing the CPUACTLR2_EL1

The CPUACTLR2_EL1 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Setting many of these bits can cause significantly lower performance on your code. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	Op0	Op1	CRn	CRm	Op2
S3_0_C15_C1_1	11	000	1111	0001	001

Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C1_1	x	x	0	-	RW	n/a	RW
S3_0_C15_C1_1	x	0	1	-	RW	RW	RW
S3_0_C15_C1_1	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Exception priority order* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state, and see *Synchronous exception prioritization* for exceptions taken to AArch64 state.

Write-Access to this register from EL1 or EL2 depends on the value of bit[0] of ACTLR_EL2 and ACTLR_EL3.

B2.40 CPUACTLR3_EL1, CPU Auxiliary Control Register 3, EL1

The CPUACTLR3_EL1 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUACTLR3_EL1 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

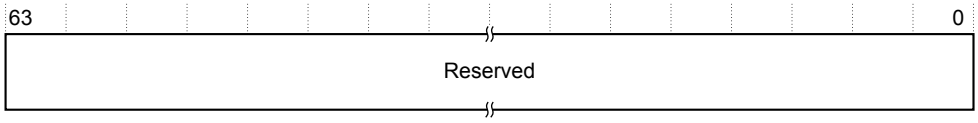

Figure B2-35 CPUACTLR3_EL1 bit assignments

Reserved, [63:0]
Reserved for Arm internal use.

Configurations

CPUACTLR3_EL1 is common to the Secure and Non-secure states.

Usage constraints

Accessing the CPUACTLR3_EL1

The CPUACTLR3_EL1 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Setting many of these bits can cause significantly lower performance on your code. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	Op0	Op1	CRn	CRm	Op2
S3_0_C15_C1_2	11	000	1111	0001	010

Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C1_2	x	x	0	-	RW	n/a	RW
S3_0_C15_C1_2	x	0	1	-	RW	RW	RW
S3_0_C15_C1_2	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Exception priority order* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state, and see *Synchronous exception prioritization* for exceptions taken to AArch64 state.

Write-Access to this register from EL1 or EL2 depends on the value of bit[0] of ACTLR_EL2 and ACTLR_EL3.

B2.41 CPUACTLR5_EL1, CPU Auxiliary Control Register 5, EL1

The CPUACTLR5_EL1 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUACTLR5_EL1 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

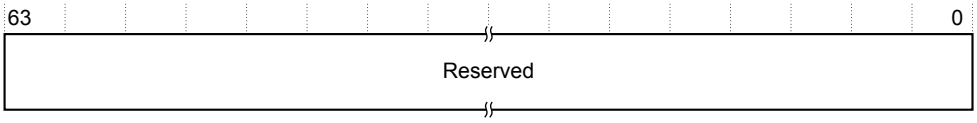

Figure B2-36 CPUACTLR5_EL1 bit assignments

Reserved, [63:0]
Reserved for Arm internal use.

Configurations

CPUACTLR5_EL1 is common to the Secure and Non-secure states.

Usage constraints

Accessing the CPUACTLR5_EL1

The CPUACTLR5_EL1 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Setting many of these bits can cause significantly lower performance on your code. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	Op0	Op1	CRn	CRm	Op2
S3_0_C15_C9_0	11	000	1111	1001	000

Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C9_0	x	x	0	-	RW	n/a	RW
S3_0_C15_C9_0	x	0	1	-	RW	RW	RW
S3_0_C15_C9_0	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Exception priority order* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state, and see *Synchronous exception prioritization* for exceptions taken to AArch64 state.

Write-Access to this register from EL1 or EL2 depends on the value of bit[0] of ACTLR_EL2 and ACTLR_EL3.

B2.42 CPUACTLR6_EL1, CPU Auxiliary Control Register 6, EL1

The CPUACTLR6_EL1 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUACTLR6_EL1 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

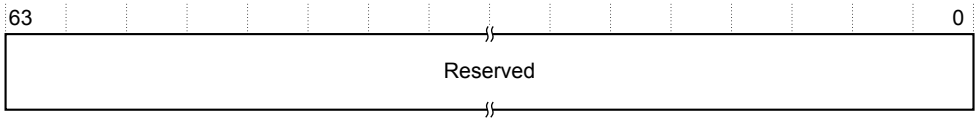

Figure B2-37 CPUACTLR6_EL1 bit assignments

Reserved, [63:0]
Reserved for Arm internal use.

Configurations

CPUACTLR6_EL1 is common to the Secure and Non-secure states.

Usage constraints

Accessing the CPUACTLR6_EL1

The CPUACTLR6_EL1 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Setting many of these bits can cause significantly lower performance on your code. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	Op0	Op1	CRn	CRm	Op2
S3_0_C15_C9_1	11	000	1111	1001	001

Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C9_1	x	x	0	-	RW	n/a	RW
S3_0_C15_C9_1	x	0	1	-	RW	RW	RW
S3_0_C15_C9_1	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Exception priority order* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state, and see *Synchronous exception prioritization* for exceptions taken to AArch64 state.

Write-Access to this register from EL1 or EL2 depends on the value of bit[0] of ACTLR_EL2 and ACTLR_EL3.

B2.43 CPUCFR_EL1, CPU Configuration Register, EL1

The CPUCFR_EL1 provides configuration information for the core.

Bit field descriptions

CPUCFR_EL1 is a 32-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

This register is read-only.

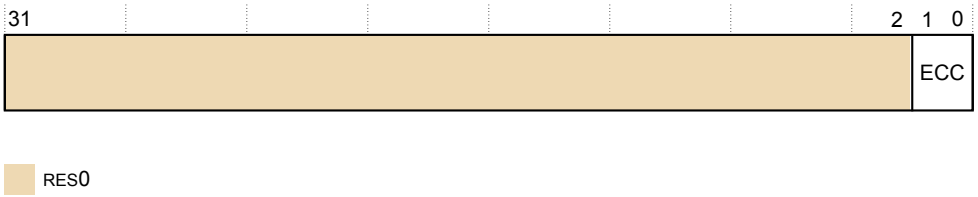

Figure B2-38 CPUCFR_EL1 bit assignments

RES0, [31:2]

RES0 Reserved

ECC, [1:0]

Indicates whether ECC is present or not. The possible values are:

00 ECC is not present.

01 ECC is present.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Usage constraints

Accessing the CPUCFR_EL1

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

To access the CPUCFR_EL1:

```
MRS <Xt>, CPUCFR_EL1 ; Read CPUCFR_EL1 into Xt
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C0_0	11	000	1111	0000	000

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C0_0	x	x	0	-	RO	n/a	RO
S3_0_C15_C0_0	x	0	1	-	RO	RO	RO
S3_0_C15_C0_0	x	1	1	-	n/a	RO	RO

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

B2.44 CPUECTLR_EL1, CPU Extended Control Register, EL1

The CPUECTLR_EL1 provides additional IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUECTLR_EL1 is a 64-bit register, and is part of the 64-bit registers functional group.

This register resets to value `0xA000000B40543000`.

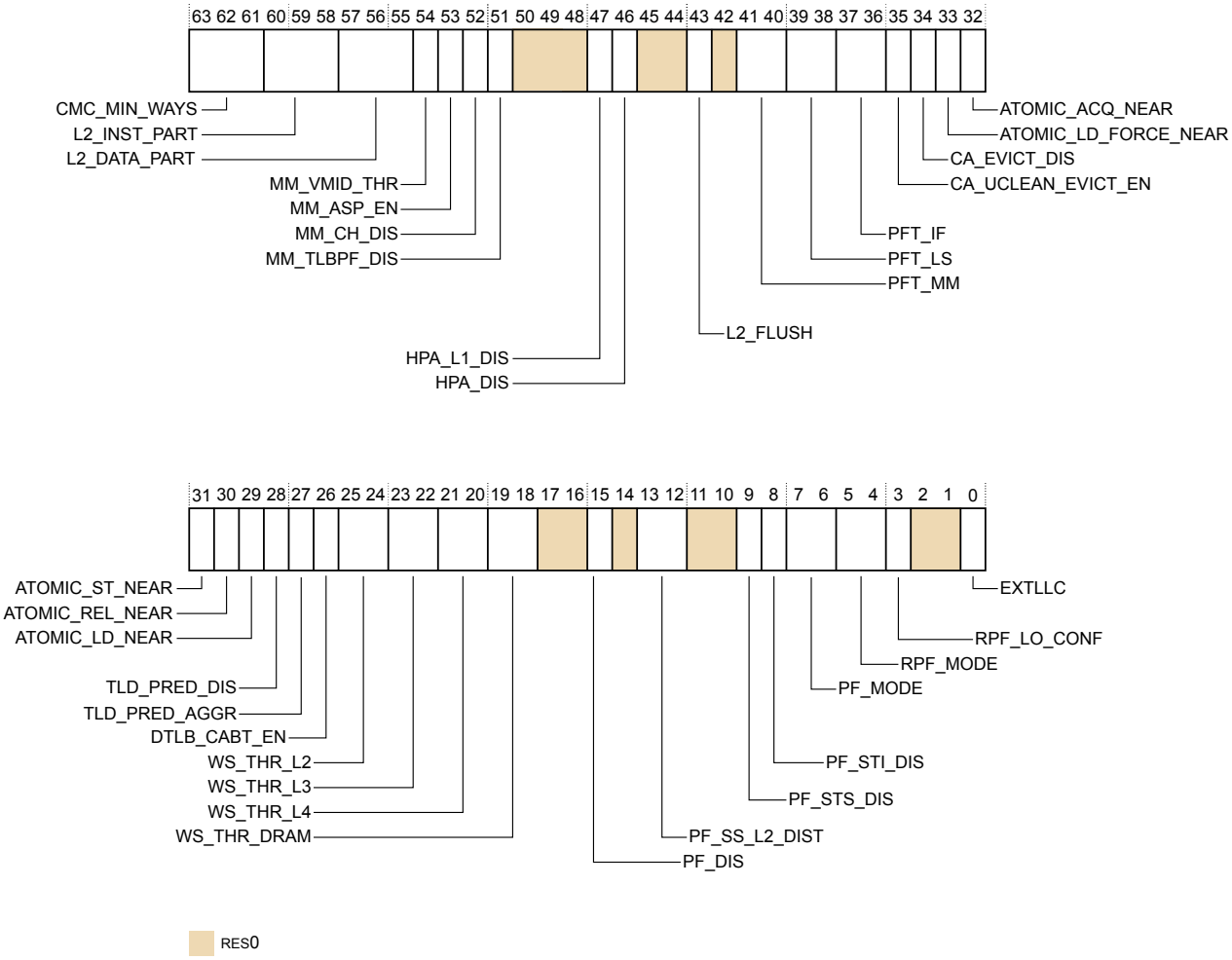

Figure B2-39 CPUECTLR_EL1 bit assignments

CMC_MIN_WAYS, [63:61]

Limits how many ways of L2 can be used by *Correlated Miss Caching* (CMC) data prefetcher. The possible values are:

`000` CMC disabled

`001` Reserved

`010` Reserved

`011` Reserved

`100` Reserved

`101` CMC must leave at least 5 ways for data in L2. This is the reset value.

110 CMC must leave at least 6 ways for data in L2.

111 CMC must leave at least 7 ways for data in L2.

L2_INST_PART, [60:58]

Partition the L2 cache for instruction.[e] The possible values are:

000 No ways reserved for instruction. This is the reset value.

001 Reserve 1 way for instructions. Only instruction fetches can allocate way [7].

010 Reserve 2 ways for instructions. Only instruction fetches can allocate ways [7:6].

011 Reserve 3 ways for instructions. Only instruction fetches can allocate ways [7:5].

100 Reserve 4 ways for instructions. Only instruction fetches can allocate ways [7:4].

101 Reserve 5 ways for instructions. Only instruction fetches can allocate ways [7:3].

110 Reserve 6 ways for instructions. Only instruction fetches can allocate ways [7:2].

111 Reserve 7 ways for instructions. Only instruction fetches can allocate ways [7:1].

L2_DATA_PART, [57:55]

Partition the L2 cache for data.[e] The possible values are:

000 No ways reserved for data. This is the reset value.

001 Reserve 1 way for data. Only data accesses can allocate way [0].

010 Reserve 2 ways for data. Only data accesses can allocate ways [1:0].

011 Reserve 3 ways for data. Only data accesses can allocate ways [2:0].

100 Reserve 4 ways for data. Only data accesses can allocate ways [3:0].

101 Reserve 5 ways for data. Only data accesses can allocate ways [4:0].

110 Reserve 6 ways for data. Only data accesses can allocate ways [5:0].

111 Reserve 7 ways for data. Only data accesses can allocate ways [6:0].

MM_VMID_THR, [54]

VMID filter threshold. The possible values are:

0 Flush VMID filter after 16 unique VMID allocations to the *Memory Management Unit* (MMU) Translation Cache. This is the reset value.

1 Flush VMID filter after 32 unique VMID allocations to the MMU Translation Cache.

MM_ASP_EN, [53]

Disables allocation of splintered pages in L2 TLB. The possible values are:

0 Enables allocation of splintered pages in the L2 TLB. This is the reset value.

1 Disables allocation of splintered pages in the L2 TLB.

MM_CH_DIS, [52]

Disables use of contiguous hint. The possible values are:

0 Enables use of contiguous hint. This is the reset value.

1 Disables use of contiguous hint.

MM_TLBPF_DIS, [51]

Disables L2 TLB prefetcher. The possible values are:

0 Enables L2 TLB prefetcher. This is the reset value.

[e] If any ways are left unselected by the settings of L2_INST_PART and L2_DATA_PART, then those ways can be used for either instruction or data allocation. If any ways selected by the settings of L2_INST_PART and L2_DATA_PART overlap, then those ways will not be used for either instruction or data allocation.

1 Disables L2 TLB prefetcher.

RES0, [50:48]

RES0 Reserved

HPA_L1_DIS, [47]

Disables HPA in L1 TLBs (but continues to use HPA in L2 TLB). The possible values are:

0 Enables hardware page aggregation in L1 TLBs. This is the reset value.

1 Disables hardware page aggregation in L1 TLBs.

HPA_DIS, [46]

Disables hardware page aggregation. The possible values are:

0 Enables hardware page aggregation. This is the reset value.

1 Disables hardware page aggregation.

RES0, [45:44]

RES0 Reserved

L2_FLUSH, [43]

Allocation behavior of copybacks caused by L2 cache hardware flush and DC CISW instructions targeting the L2 cache. If it is known that data is likely to be used soon by another core, setting this bit can improve system performance. The possible values are:

0 L2 cache flushes and invalidates by set/way do not allocate in the L3 cache. Cache lines in the UniqueDirty state cause Write-Back transactions with the allocation hint cleared, while cache lines in UniqueClean or SharedClean states cause address-only Evict transactions. This is the reset value.

1 L2 cache flushes by set/way allocate in the L3 cache. Cache lines in the UniqueDirty or UniqueClean state cause WriteBackFull or WriteEvictFull transactions, respectively, both with the allocation hint set. Cache lines in the SharedClean state cause address-only Evict transactions.

RES0, [42]

RES0 Reserved

PFT_MM, [41:40]

DRAM prefetch using PrefetchTgt transactions for table walk requests. The possible values are:

00 Disable PrefetchTgt generation for requests from the MMU. This is the reset value.

01 Conservatively generate PrefetchTgt for cacheable requests from the MMU, always generate for Non-cacheable.

10 Aggressively generate PrefetchTgt for cacheable requests from the MMU, always generate for Non-cacheable.

11 Always generate PrefetchTgt for cacheable requests from the MMU, always generate for Non-cacheable.

PFT_LS, [39:38]

DRAM prefetch using PrefetchTgt transactions for load and store requests. The possible values are:

00 Disable PrefetchTgt generation for requests from the Load-Store unit (LS). This is the reset value.

01 Conservatively generate PrefetchTgt for cacheable requests from the LS, always generate for Non-cacheable.

10 Aggressively generate PrefetchTgt for cacheable requests from the LS, always generate for Non-cacheable.

11 Always generate PrefetchTgt for cacheable requests from the LS, always generate for Non-cacheable.

PFT_IF, [37:36]

DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. The possible values are:

00 Disable PrefetchTgt generation for requests from the Instruction Fetch unit (IF). This is the reset value.

01 Conservatively generate PrefetchTgt for cacheable requests from the IF, always generate for Non-cacheable.

10 Aggressively generate PrefetchTgt for cacheable requests from the IF, always generate for Non-cacheable.

11 Always generate PrefetchTgt for cacheable requests from the IF, always generate for Non-cacheable.

CA_UCLEAN_EVICT_EN, [35]

Enables sending WriteEvict transactions on the CPU CHI interface for UniqueClean evictions. WriteEvict transactions update downstream caches. Enable WriteEvict transactions only if there is an additional level of cache below the CPU's L2 cache. The possible values are:

0 Disables sending data with UniqueClean evictions.

1 Enables sending data with UniqueClean evictions. This is the reset value.

CA_EVICT_DIS, [34]

Disables sending of Evict transactions on the CPU CHI interface for clean cache lines that are evicted from the core. Evict transactions are required only if the system contains a snoop filter that requires notification when the core evicts the cache line. The possible values are:

0 Enables sending Evict transactions. This is the reset value.

1 Disables sending Evict transactions.

ATOMIC_LD_FORCE_NEAR, [33]

A load atomic (including SWP and CAS) instruction to WB memory will be performed near. The possible values are:

0 Load-atomic is near if cache line is already exclusive, otherwise make far atomic request.

1 Load-atomic will be performed near by bringing the line into the L1D Cache. This is the reset value.

ATOMIC_ACQ_NEAR, [32]

An atomic instruction to WB memory with acquire semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:

0 Acquire-atomic is near if cache line is already Exclusive, otherwise make far atomic request.

1 Acquire-atomic will make up to 1 fill request to perform near. This is the reset value.

ATOMIC_ST_NEAR, [31]

A store atomic instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:

`0` Store-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the reset value.

`1` Store-atomic will make up to 1 fill request to perform near.

ATOMIC_REL_NEAR, [30]

An atomic instruction to WB memory with release semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:

`0` Release-atomic is near if cache line is already Exclusive, otherwise make far atomic request.

`1` Release-atomic will make up to 1 fill request to perform near. This is the reset value.

ATOMIC_LD_NEAR, [29]

A load atomic (including SWP and CAS) instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:

`0` Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the reset value.

`1` Load-atomic will make up to 1 fill request to perform near.

TLD_PRED_DIS, [28]

Disables Transient Load Prediction. The possible values are:

`0` Enables transient load prediction. This is the reset value.

`1` Disables transient load prediction.

TLD_PRED_AGGR, [27]

Enables aggressive transient load prediction. The possible values are:

`0` Transient load prediction uses more conservative threshold. This is the reset value.

`1` Transient load prediction uses more aggressive threshold.

DTLB_CABT_EN, [26]

Enables TLB Conflict Data Abort Exception. The possible values are:

`0` Disables TLB conflict data abort exception. This is the reset value.

`1` Enables TLB conflict data abort exception.

WS_THR_L2, [25:24]

Threshold for direct stream to L2 cache on store. The possible values are:

`00` 256B. This is the reset value.

`01` 4KB

`10` 8KB

`11` Disables direct stream to L2 cache on store.

WS_THR_L3, [23:22]

Threshold for direct stream to L3 cache on store. Once the threshold is met, consecutive streaming writes will not allocate in the L1 or L2 cache. The possible values are:

`00` 128KB

`01` 256KB. This is the reset value.

`10` 512KB

`11` Disables direct stream to L3 cache on store.

WS_THR_L4, [21:20]

Threshold for direct stream to L4 cache on store. Once the threshold is met, consecutive streaming writes will not allocate in the L1, L2, or L3 cache. [f] The possible values are:

`00` 256KB

`01` 512KB. This is the reset value.

`10` 1MB

`11` Disables direct stream to L4 cache on store.

WS_THR_DRAM, [19:18]

Threshold for direct stream to DRAM on store.[f] The possible values are:

`00` 512KB

`01` 1MB. This is the reset value.

`10` 2MB

`11` Disables direct stream to DRAM on store.

RES0, [17:16]

RES0 Reserved

PF_DIS, [15]

Disables data-side hardware prefetching. The possible values are:

`0` Enables hardware prefetching. This is the reset value.

`1` Disables hardware prefetching.

RES0, [14]

RES0 Reserved

PF_SS_L2_DIST, [13:12]

Single cache line stride prefetching L2 distance. The possible values are:

`00` 22

`01` 40

`10` 60

`11` Dynamically managed by hardware. This is the reset value.

RES0, [11:10]

RES0 Reserved

PF_STS_DIS, [9]

Disable store-stride prefetches. The possible values are:

`0` Enables store prefetching. This is the reset value.

`1` Disables store prefetching.

PF_STI_DIS, [8]

Disables store prefetches at issue. The possible values are:

[f] If the PE detects that it is the only active requestor in the DSU cluster, then it may scale these to larger thresholds.

`0` Enables store prefetching. This is the reset value.

`1` Disables store prefetching.

PF_MODE, [7:6]

General prefetcher aggressibility. The possible values are:

`00` Dynamic aggressiveness. This is the reset value.

`01` Conservative prefetching

`10` Very conservative prefetching

`11` Most conservative prefetching

RPF_MODE, [5:4]

Region prefetcher aggressibility. The possible values are:

`00` Dynamic region prefetch aggressiveness. This is the reset value.

`01` Conservative region prefetching

`10` Very conservative region prefetching

`11` Most conservative region prefetching. This will disable the region prefetcher.

RPF_LO_CONF, [3]

Region prefetcher single accesses training behavior. The possible values are:

`0` Limited training for PHT on single accesses. This is the reset value.

`1` Always train the PHT on single accesses, which results in fewer prefetch requests.

RES0, [2:1]

RES0 Reserved

EXTLLC, [0]

Internal or external Last-level cache (LLC) in the system. The possible values are:

`0` Indicates that an internal Last-level cache is present in the system, and that the DataSource field on the master CHI interface indicates when data is returned from the LLC. This is used to control how the LL_CACHE* PMU events count. This is the reset value.

`1` Indicates that an external Last-level cache is present in the system, and that the DataSource field on the master CHI interface indicates when data is returned from the LLC. This is used to control how the LL_CACHE* PMU events count. Additionally, the core makes performance optimizations based on the LLC DataSource.

Configurations

This register has no configuration options.

Usage constraints

Accessing the CPUECTLR_EL1

The CPU Extended Control Register can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
CPUECTLR_EL1	11	000	1111	0001	100

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CPUECTLR_EL1	x	x	0	-	RW	n/a	RW
CPUECTLR_EL1	x	0	1	-	RW	RW	RW
CPUECTLR_EL1	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

Access to this register depends on bit[1] of ACTLR_EL2 and ACTLR_EL3.

B2.45 CPUECTLR2_EL1, CPU Extended Control Register2, EL1

The CPUECTLR2_EL1 provides additional IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUECTLR2_EL1 is a 64-bit register, and is part of the 64-bit registers functional group.

This register resets to value `0x0000000000000000`.

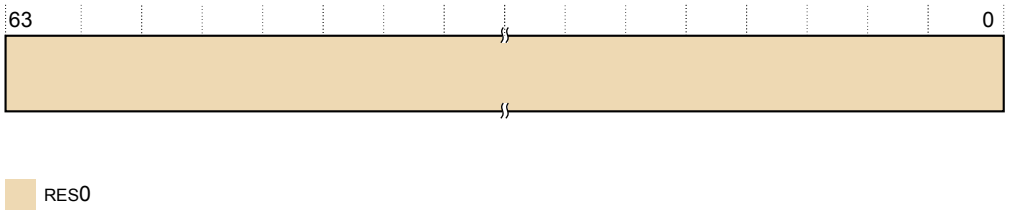

Figure B2-40 CPUECTLR2_EL1 bit assignments

RES0, [63:0]

RES0 Reserved

Configurations

This register has no configuration options.

Usage constraints

Accessing the CPUECTLR2_EL1

The CPU Extended Control Register can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
CPUECTLR2_EL1	11	000	1111	0001	101

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CPUECTLR2_EL1	x	x	0	-	RW	n/a	RW
CPUECTLR2_EL1	x	0	1	-	RW	RW	RW
CPUECTLR2_EL1	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

Access to this register depends on bit[1] of ACTLR_EL2 and ACTLR_EL3.

B2.46 CPUPCR_EL3, CPU Private Control Register, EL3

The CPUPCR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPCR_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

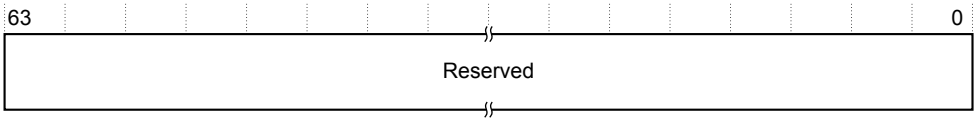

Figure B2-41 CPUPCR_EL3 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPCR_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPCR_EL3

The CPUPCR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_1	11	110	1111	1000	001

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_1	x	x	0	-	-	n/a	RW
S3_6_C15_8_1	x	0	1	-	-	-	RW
S3_6_C15_8_1	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.47 CPUPFR_EL3, CPU Private Flag Register, EL3

The CPUPCR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPCR_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

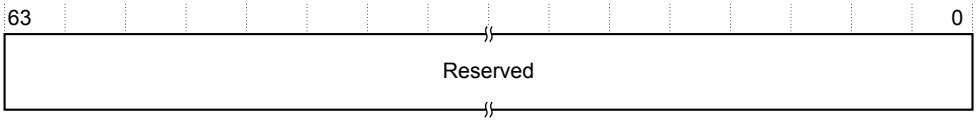

Figure B2-42 CPUPCR_EL3 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPCR_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPCR_EL3

The CPUPCR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_6	11	110	1111	1000	001

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_6	x	x	0	-	-	n/a	RW
S3_6_C15_8_6	x	0	1	-	-	-	RW
S3_6_C15_8_6	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.48 CPUPMR_EL3, CPU Private Mask Register, EL3

The CPUPMR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPMR_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

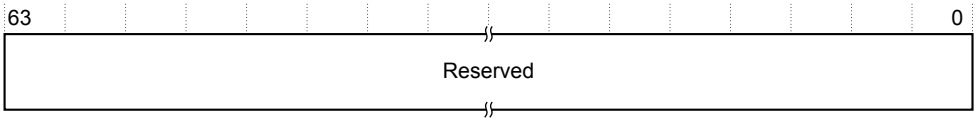

Figure B2-43 CPUPMR_EL3 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPMR_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPMR_EL3

The CPUPMR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_3	11	110	1111	1000	011

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_3	x	x	0	-	-	n/a	RW
S3_6_C15_8_3	x	0	1	-	-	-	RW
S3_6_C15_8_3	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.49 CPUPMR2_EL3, CPU Private Mask Register 2, EL3

The CPUPMR2_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPMR2_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

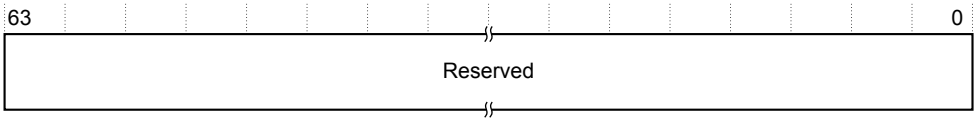

Figure B2-44 CPUPMR2_EL3 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPMR2_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPMR2_EL3

The CPUPMR2_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_5	11	110	1111	1000	011

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_5	x	x	0	-	-	n/a	RW
S3_6_C15_8_5	x	0	1	-	-	-	RW
S3_6_C15_8_5	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.50 CPUPOR_EL3, CPU Private Operation Register, EL3

The CPUPOR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPOR_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

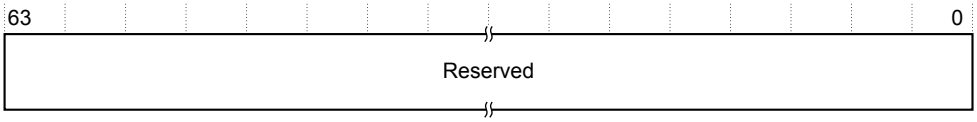

Figure B2-45 CPUPOR_EL3 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPOR_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPOR_EL3

The CPUPOR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_2	11	110	1111	1000	010

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_2	x	x	0	-	-	n/a	RW
S3_6_C15_8_2	x	0	1	-	-	-	RW
S3_6_C15_8_2	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.51 CPUPOR2_EL3, CPU Private Operation Register 2, EL3

The CPUPOR2_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPOR2_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

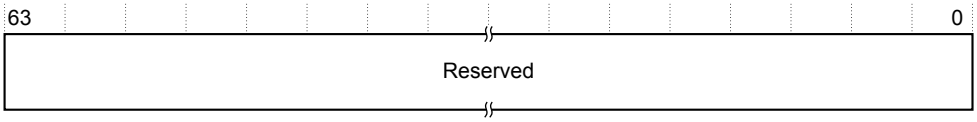

Figure B2-46 CPUPOR2_EL3 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPOR2_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPOR2_EL3

The CPUPOR2_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_4	11	110	1111	1000	010

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_4	x	x	0	-	-	n/a	RW
S3_6_C15_8_4	x	0	1	-	-	-	RW
S3_6_C15_8_4	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.52 CPUPPMCR_EL3, CPU Power Performance Management Configuration Register, EL3

The CPUPPMCR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPPMCR_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

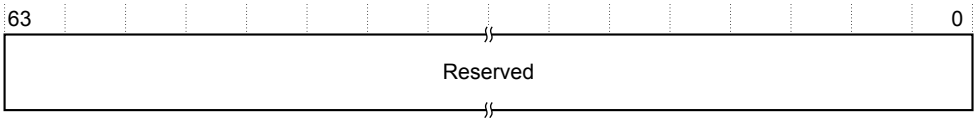

Figure B2-47 CPUPPMCR_EL3 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPPMCR_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPPMCR_EL3

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_0	11	110	1111	0010	000

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_C2_0	x	x	0	-	-	n/a	RW
S3_6_C15_C2_0	x	0	1	-	-	-	RW
S3_6_C15_C2_0	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.53 CPUPSELR_EL3, CPU Private Selection Register, EL3

The CPUPSELR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPSELR_EL3 is a 32-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

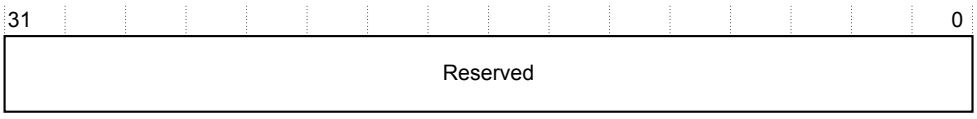

Figure B2-48 CPUPSELR_EL3 bit assignments

Reserved, [31:0]

Reserved for Arm internal use.

Configurations

CPUPSELR_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPSELR_EL3

The CPUPSELR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_8_0	11	110	1111	1000	000

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_6_C15_8_0	x	x	0	-	-	n/a	RW
S3_6_C15_8_0	x	0	1	-	-	-	RW
S3_6_C15_8_0	x	1	1	-	n/a	-	RW

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.54 CPUPWRCTLR_EL1, Power Control Register, EL1

The CPUPWRCTLR_EL1 provides information about power control support for the core.

Bit field descriptions

CPUPWRCTLR_EL1 is a 32-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

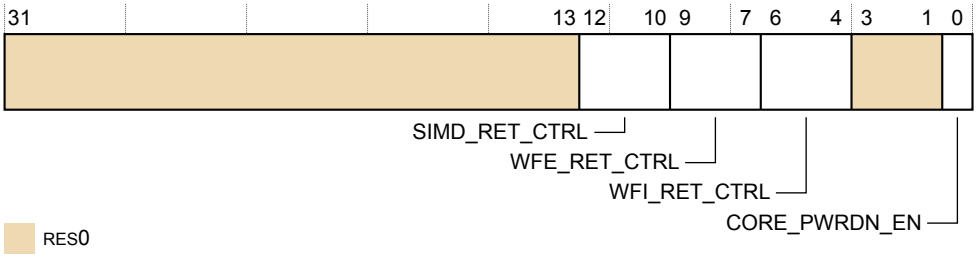

Figure B2-49 CPUPWRCTLR_EL1 bit assignments

RES0, [31:10]

RES0 Reserved

SIMD_RET_CTRL, [12:10]

Advanced SIMD and floating-point retention control:

000 Disable the retention circuit. This is the default value, see *Table B2-17 CPUPWRCTLR Retention Control Field* on page B2-232 for more retention control options.

WFE_RET_CTRL, [9:7]

CPU WFE retention control:

000 Disable the retention circuit. This is the default value, see *Table B2-17 CPUPWRCTLR Retention Control Field* on page B2-232 for more retention control options.

WFI_RET_CTRL, [6:4]

CPU WFI retention control:

000 Disable the retention circuit. This is the default value, see *Table B2-17 CPUPWRCTLR Retention Control Field* on page B2-232 for more retention control options.

RES0, [3:1]

RES0 Reserved

CORE_PWRDN_EN, [0]

Indicates to the power controller using PACTIVE if the core wants to power down when it enters WFI state.

0 No power down requested. This is the reset value.

1 A power down is requested.

Table B2-17 CPUPWRCTLR Retention Control Field

Encoding	Number of counter ticks[g]	Minimum retention entry delay (System counter at 50MHz-10MHz)
000	Disable the retention circuit	Default Condition
001	2	40ns-200ns
010	8	160ns-800ns
011	32	640ns – 3,200ns
100	64	1,280ns-6,400ns
101	128	2,560ns-12,800ns
110	256	5,120ns-25,600ns
111	512	10,240ns-51,200ns

Configurations

There are no configuration notes.

Usage constraints

Accessing the CPUPWRCTLR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_7	11	000	1111	0010	111

Accessibility

This register is accessible in software as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C2_7	x	x	0	-	RW	n/a	RW
S3_0_C15_C2_7	x	0	1	-	RW	RW	RW
S3_0_C15_C2_7	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

[g] The number of system counter ticks required before the core signals retention readiness on PACTIVE to the power controller. The core does not accept a retention entry request until this time.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

Write-Access to this register from EL1 or EL2 depends on the value of bit[7] of ACTLR_EL2 and ACTLR_EL3.

B2.55 CSSELR_EL1, Cache Size Selection Register, EL1

CSSELR_EL1 selects the current Cache Size ID Register (CCSIDR_EL1), by specifying:

- The required cache level
- The cache type, either instruction or data cache

For details of the CCSIDR_EL1, see *B2.33 CCSIDR_EL1, Cache Size ID Register, EL1* on page B2-186.

Bit field descriptions

CSSELR_EL1 is a 32-bit register, and is part of the Identification registers functional group.

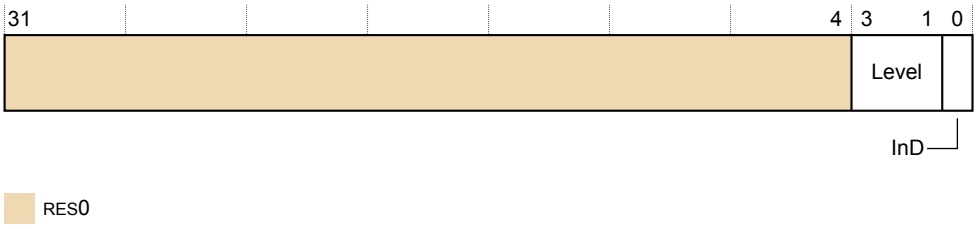

Figure B2-50 CSSELR_EL1 bit assignments

RES0, [31:4]

RES0 Reserved

Level, [3:1]

Cache level of required cache:

`000` L1

`001` L2

`010` L3, if present

The combination of Level=`001` and InD=`1` is reserved.

The combinations of Level and InD for `0100` to `1111` are reserved.

InD, [0]

Instruction not Data bit:

`0` Data or unified cache

`1` Instruction cache

The combination of Level=`001` and InD=`1` is reserved.

The combinations of Level and InD for `0100` to `1111` are reserved.

Configurations

If a cache level is missing but CSSELR_EL1 selects this level, then a CCSIDR_EL1 read returns an UNKNOWN value.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.56 CTR_EL0, Cache Type Register, EL0

The CTR_EL0 provides information about the architecture of the caches.

Bit field descriptions

CTR_EL0 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

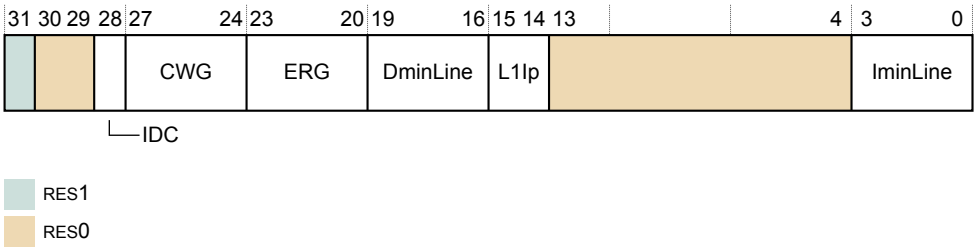

Figure B2-51 CTR_EL0 bit assignments

RES1, [31]

RES1 Reserved

RES0, [30:29]

RES0 Reserved

IDC, [28]

Data cache clean requirements for instruction to data coherence:

0 Data cache clean to the point of unification is required for instruction to data coherence, unless CLIDR_EL1.LoC == 0b000 or (CLIDR_EL1.LoUIS == 0b000 && CLIDR_EL1.LoUU == 0b000). .

1 Data cache clean to the point of unification is not required for instruction to data coherence.

IDC reflects the inverse value of the **BROADCASTCACHEMAINTPOU** pin.

CWG, [27:24]

Cache write-back 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:

0100 Cache write-back granule size is 16 words.

ERG, [23:20]

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:

0100 Exclusive reservation granule size is 16 words.

DminLine, [19:16]

Log_2 of the number of words in the smallest cache line of all the data and unified caches that the core controls:

0100 Smallest data cache line size is 16 words.

L1Ip, [15:14]

Instruction cache policy. Indicates the indexing and tagging policy for the L1 Instruction cache:

`11` *Physically Indexed Physically Tagged* (PIPT).

RES0, [13:4]

RES0 Reserved

IminLine, [3:0]

Log_2 of the number of words in the smallest cache line of all the instruction caches that the core controls.

`0100` Smallest instruction cache line size is 16 words.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.57 DCZID_EL0, Data Cache Zero ID Register, EL0

The DCZID_EL0 indicates the block size written with byte values of zero by the `DC ZVA` (Data Cache Zero by Address) system instruction.

Bit field descriptions

DCZID_EL0 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

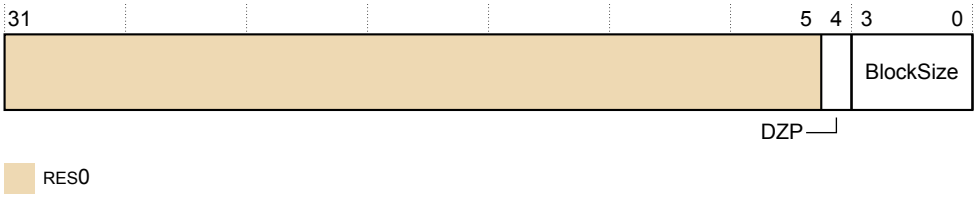

Figure B2-52 DCZID_EL0 bit assignments

RES0, [31:5]

RES0 Reserved

BlockSize, [3:0]

Log_2 of the block size in words:

`0100` The block size is 16 words.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.58 DISR_EL1, Deferred Interrupt Status Register, EL1

The DISR_EL1 records the SError interrupts consumed by an `ESB` instruction.

Bit field descriptions

DISR_EL1 is a 64-bit register, and is part of the registers *Reliability, Availability, Serviceability* (RAS) functional group.

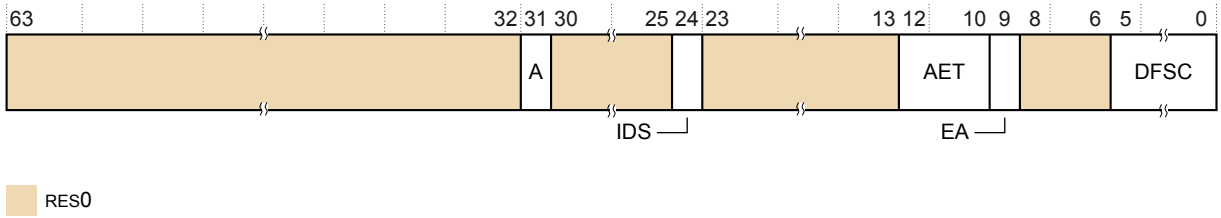

Figure B2-53 DISR_EL1 bit assignments, DISR_EL1.IDS is 0

RES0, [63:32]

RES0 Reserved

A, [31]

Set to 1 when ESB defers an asynchronous SError interrupt. If the implementation does not include any synchronizable sources of SError interrupt, this bit is RES0.

RES0, [30:25]

RES0 Reserved

IDS, [24]

Indicates the type of format the deferred SError interrupt uses. The value of this bit is:

`0` Deferred error uses architecturally-defined format.

RES0, [23:13]

RES0 Reserved

AET, [12:10]

Asynchronous Error Type. Describes the state of the core after taking an asynchronous Data Abort exception. The possible values are:

`000` Uncontainable error (UC)

`001` Unrecoverable error (UEU)

Note

The recovery software must also examine any implemented fault records to determine the location and extent of the error.

EA, [9]

RES0 Reserved

RES0, [8:6]

RES0 Reserved

DFSC, [5:0]

Data Fault Status Code. The possible values of this field are:

`010001` Asynchronous SError interrupt

Note

In AArch32 the `010001` code previously meant an Asynchronous External Abort on memory access. With the RAS extension, it extends to include any asynchronous SError interrupt. The Parity Error codes are not used in the RAS extension.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.59 ERRIDR_EL1, Error ID Register, EL1

The ERRIDR_EL1 defines the number of error record registers.

Bit field descriptions

ERRIDR_EL1 is a 32-bit register, and is part of the registers *Reliability, Availability, Serviceability* (RAS) functional group.

This register is read-only.

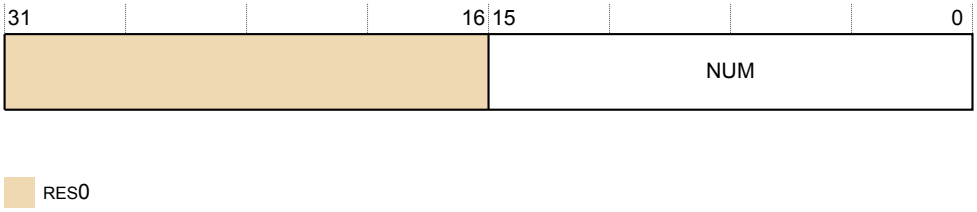

Figure B2-54 ERRIDR_EL1 bit assignments

RES0, [31:16]

RES0 Reserved

NUM, [15:0]

Number of records that can be accessed through the Error Record system registers.

`0x0002` Two records present, if L3 cache is present.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.60 ERRSELR_EL1, Error Record Select Register, EL1

The ERRSELR_EL1 selects which error record should be accessed through the Error Record system registers. This register is not reset on a Warm reset.

Bit field descriptions

ERRSELR_EL1 is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

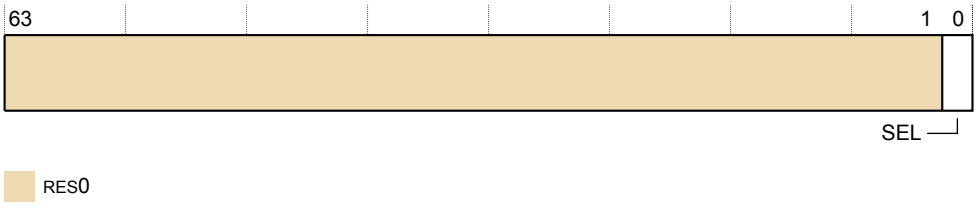

Figure B2-55 ERRSELR_EL1 bit assignments

RES0, [63:1]

RES0 Reserved

SEL, [0]

Selects which error record should be accessed.

`0` Select error record 0 containing errors from L1 and L2 RAMs located on the Cortex-A78C core.

`1` Select error record 1 containing errors from L3 RAMs located on the DSU.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.61 ERXADDR_EL1, Selected Error Record Address Register, EL1

Register ERXADDR_EL1 accesses the ERR<n>ADDR address register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXADDR_EL1 accesses the ERR0ADDR register of the core error record. See *B3.2 ERR0ADDR, Error Record Address Register* on page B3-345.

If ERRSELR_EL1.SEL==1, then ERXADDR_EL1 accesses the ERR1ADDR register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

B2.62 ERXCTLR_EL1, Selected Error Record Control Register, EL1

Register ERXCTLR_EL1 accesses the ERR<n>CTLR control register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXCTLR_EL1 accesses the ERR0CTLR register of the core error record. See *B3.3 ERR0CTLR, Error Record Control Register* on page B3-346.

If ERRSELR_EL1.SEL==1, then ERXCLTR_EL1 accesses the ERR1CTLR register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

B2.63 ERXFR_EL1, Selected Error Record Feature Register, EL1

Register ERXFR_EL1 accesses the ERR<n>FR feature register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXFR_EL1 accesses the ERR0FR register of the core error record. See *B3.4 ERR0FR, Error Record Feature Register* on page B3-348.

If ERRSELR_EL1.SEL==1, then ERXFR_EL1 accesses the ERR1FR register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

B2.64 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1

Register ERXMISC0_EL1 accesses the ERR<n>MISC0 register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXMISC0_EL1 accesses the ERR0MISC0 register of the core error record. See *B3.5 ERR0MISC0, Error Record Miscellaneous Register 0* on page B3-350.

If ERRSELR_EL1.SEL==1, then ERXMISC0_EL1 accesses the ERR1MISC0 register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

B2.65 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1, EL1

Register ERXMISC1_EL1 accesses the ERR<n>MISC1 miscellaneous register 1 for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXMISC1_EL1 accesses the ERR0MISC1 register of the core error record. See *B3.6 ERR0MISC1, Error Record Miscellaneous Register 1* on page B3-355.

If ERRSELR_EL1.SEL==1, then ERXMISC1_EL1 accesses the ERR1MISC1 register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

B2.66 ERXPFGCDNR_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1

Register ERXPFGCDNR_EL1 accesses the ERR<n>PFGCNDR register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXPFGCDNR_EL1 accesses the ERR0PFGCDNR register of the core error record. See *B3.7 ERR0PFGCDNR, Error Pseudo Fault Generation Count Down Register* on page B3-356.

If ERRSELR_EL1.SEL==1, then ERXPFGCDNR_EL1 accesses the ERR1PFGCDNR register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

Configurations

There are no configuration notes.

Accessing the ERXPFGCDNR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <Xt>,<systemreg>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_2	11	000	1111	0010	010

Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C2_2	x	x	0	-	RW	n/a	RW
S3_0_C15_C2_2	x	0	1	-	RW	RW	RW
S3_0_C15_C2_2	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. Executing the PE at this Exception level is not permitted.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Exception priority order* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch32 state, and see *Synchronous exception prioritization* for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGCDNR_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR_EL2 and ACTLR_EL3. See *B2.6 ACTLR_EL2, Auxiliary Control Register, EL2* on page B2-152 and *B2.7 ACTLR_EL3, Auxiliary Control Register, EL3* on page B2-155.

ERXPFGCDNR_EL1 is UNDEFINED at EL0.

B2.67 ERXPFGCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1

Register ERXPFGCTLR_EL1 accesses the ERR<n>PFGCTLR register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXPFGCTLR_EL1 accesses the ERR0PFGCTLR register of the core error record. See *B3.8 ERR0PFGCTLR, Error Pseudo Fault Generation Control Register* on page B3-357.

If ERRSELR_EL1.SEL==1, then ERXPFGCTLR_EL1 accesses the ERR1PFGCTLR register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

Configurations

There are no configuration notes.

Accessing the ERXPFGCTLR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written using MSR with the following syntax:

```
MSR <Xt>,<systemreg>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_1	11	000	1111	0010	001

Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C2_1	x	x	0	-	RW	n/a	RW
S3_0_C15_C2_1	x	0	1	-	RW	RW	RW
S3_0_C15_C2_1	x	1	1	-	n/a	RW	RW

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Exception priority order* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch32 state, and see *Synchronous exception prioritization* for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGCTLR_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR_EL2 and ACTLR_EL3. See *B2.6 ACTLR_EL2, Auxiliary Control Register, EL2* on page B2-152 and *B2.7 ACTLR_EL3, Auxiliary Control Register, EL3* on page B2-155.

ERXPFGCTLR_EL1 is UNDEFINED at EL0.

If ERXPFGCTLR_EL1 is accessible at EL1 and HCR_EL2.TERR == 1, then direct reads and writes of ERXPFGCTLR_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGCTLR_EL1 is accessible at EL1 or EL2 and SCR_EL3.TERR == 1, then direct reads and writes of ERXPFGCTLR_EL1 at EL1 or EL2 generate a Trap exception to EL3.

B2.68 ERXPFGFR_EL1, Selected Pseudo Fault Generation Feature Register, EL1

Register ERXPFGFR_EL1 accesses the ERR<n>PFGFR register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXPFGFR_EL1 accesses the ERR0PFGFR register of the core error record. See *B3.9 ERR0PFGFR, Error Pseudo Fault Generation Feature Register* on page B3-361.

If ERRSELR_EL1.SEL==1, then ERXPFGFR_EL1 accesses the ERR1PFGFR register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

Configurations

This core has no configuration notes.

Accessing the ERXPFGFR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>,<systemreg>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_0	11	000	1111	0010	000

Accessibility

This register is accessible in software as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
S3_0_C15_C2_0	x	x	0	-	RO	n/a	RO
S3_0_C15_C2_0	x	0	1	-	RO	RO	RO
S3_0_C15_C2_0	x	1	1	-	n/a	RO	RO

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see *Exception priority order* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch32 state, and see *Synchronous exception prioritization* for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGR_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR_EL2 and ACTLR_EL3. See *B2.6 ACTLR_EL2, Auxiliary Control Register, EL2* on page B2-152 and *B2.7 ACTLR_EL3, Auxiliary Control Register, EL3* on page B2-155.

ERXPFGR_EL1 is UNDEFINED at EL0.

If ERXPFGR_EL1 is accessible at EL1 and HCR_EL2.TERR == 1, then direct reads and writes of ERXPFGR_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGR_EL1 is accessible at EL1 or EL2 and SCR_EL3.TERR == 1, then direct reads and writes of ERXPFGR_EL1 at EL1 or EL2 generate a Trap exception to EL3.

B2.69 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1

Register ERXSTATUS_EL1 accesses the ERR<n>STATUS primary status register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXSTATUS_EL1 accesses the ERR0STATUS register of the core error record. See *B3.10 ERR0STATUS, Error Record Primary Status Register* on page B3-364.

If ERRSELR_EL1.SEL==1, then ERXSTATUS_EL1 accesses the ERR1STATUS register of the DSU error record. See the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

B2.70 ESR_EL1, Exception Syndrome Register, EL1

The ESR_EL1 holds syndrome information for an exception taken to EL1.

Bit field descriptions

ESR_EL1 is a 32-bit register, and is part of the Exception and fault handling registers functional group.

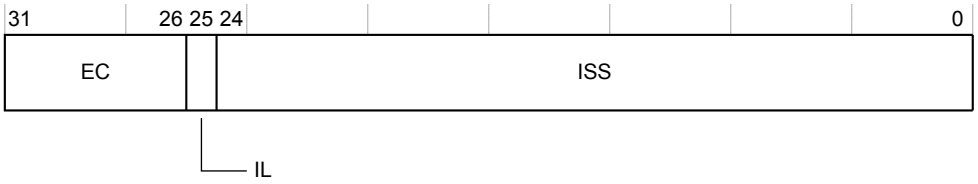

Figure B2-56 ESR_EL1 bit assignments

EC, [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

IL, [25]

Instruction Length for synchronous exceptions. The possible values are:

`0` 16-bit

`1` 32-bit

This field is `1` for the SError interrupt, instruction aborts, misaligned PC, Stack pointer misalignment, data aborts for which the ISV bit is `0`, exceptions caused by an illegal instruction set state, and exceptions using the `0x00` Exception Class.

ISS, [24:0]

Syndrome information

When reporting a virtual SEI, bits[24:0] take the value of VSESRL_EL2[24:0].

When reporting a physical SEI, the following occurs:

- IDS==0 (architectural syndrome)
- AET always reports an uncontainable error (UC) with value `0b000` or an unrecoverable error (UEU) with value `0b001`.
- EA is RES0.

When reporting a synchronous data abort, EA is RES0.

See *B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B2-339.

Configurations

This register has no configuration options.

B2.71 ESR_EL2, Exception Syndrome Register, EL2

The ESR_EL2 holds syndrome information for an exception taken to EL2.

Bit field descriptions

ESR_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group
- The Exception and fault handling registers functional group

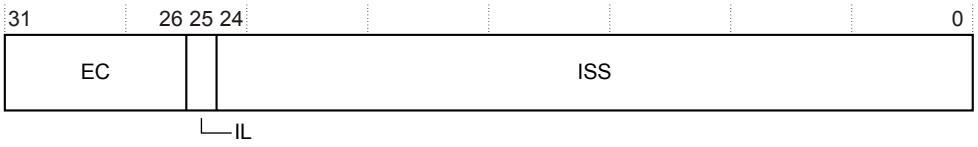

Figure B2-57 ESR_EL2 bit assignments

EC, [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

IL, [25]

Instruction Length for synchronous exceptions. The possible values are:

`0` 16-bit

`1` 32-bit

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

ISS, [24:0]

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

When reporting a virtual SEI, bits[24:0] take the value of VSESRL_EL2[24:0].

When reporting a physical SEI, the following occurs:

- IDS==0 (architectural syndrome)
- AET always reports an uncontainable error (UC) with value `0b000` or an unrecoverable error (UEU) with value `0b001`.
- EA is RES0.

When reporting a synchronous Data Abort, EA is RES0.

See *B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B2-339.

Configurations

RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.72 ESR_EL3, Exception Syndrome Register, EL3

The ESR_EL3 holds syndrome information for an exception taken to EL3.

Bit field descriptions

ESR_EL3 is a 32-bit register, and is part of the Exception and fault handling registers functional group.

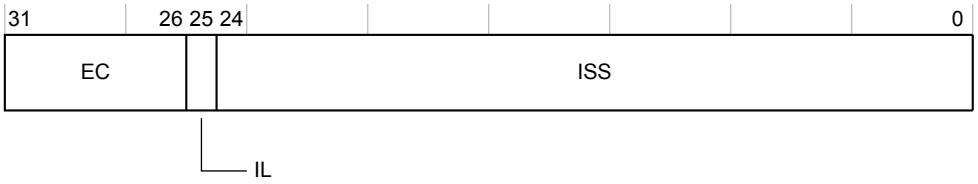

Figure B2-58 ESR_EL3 bit assignments

EC, [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

IL, [25]

Instruction Length for synchronous exceptions. The possible values are:

`0` 16-bit

`1` 32-bit

This field is `1` for the SError interrupt, instruction aborts, misaligned PC, Stack pointer misalignment, data aborts for which the ISV bit is `0`, exceptions caused by an illegal instruction set state, and exceptions using the `0x0` Exception Class.

ISS, [24:0]

Syndrome information

When reporting a virtual SEI, bits[24:0] take the value of VSESRL_EL2[24:0].

When reporting a physical SEI, the following occurs:

- IDS==0 (architectural syndrome)
- AET always reports an uncontainable error (UC) with value `0b000` or an unrecoverable error (UEU) with value `0b001`.
- EA is RES0.

When reporting a synchronous data abort, EA is RES0.

See *B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B2-339.

Configurations

RW fields in this register reset to architecturally UNKNOWN values.

B2.73 HACR_EL2, Hyp Auxiliary Configuration Register, EL2

HACR_EL2 controls trapping to EL2 of IMPLEMENTATION DEFINED aspects of Non-secure EL1 or EL0 operation. This register is not used in the Cortex-A78C core.

Bit field descriptions

HACR_EL2 is a 32-bit register, and is part of Virtualization registers functional group.

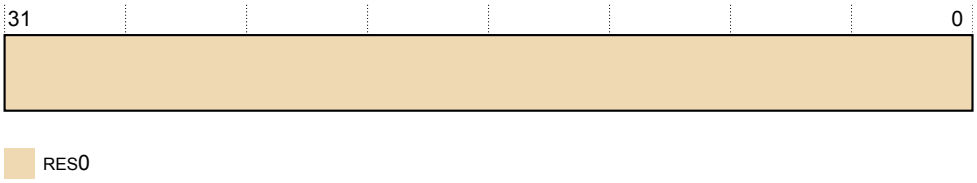

Figure B2-59 HACR_EL2 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.74 HCR_EL2, Hypervisor Configuration Register, EL2

The HCR_EL2 provides configuration control for virtualization, including whether various Non-secure operations are trapped to EL2.

Bit field descriptions

HCR_EL2 is a 64-bit register, and is part of the Virtualization registers functional group.

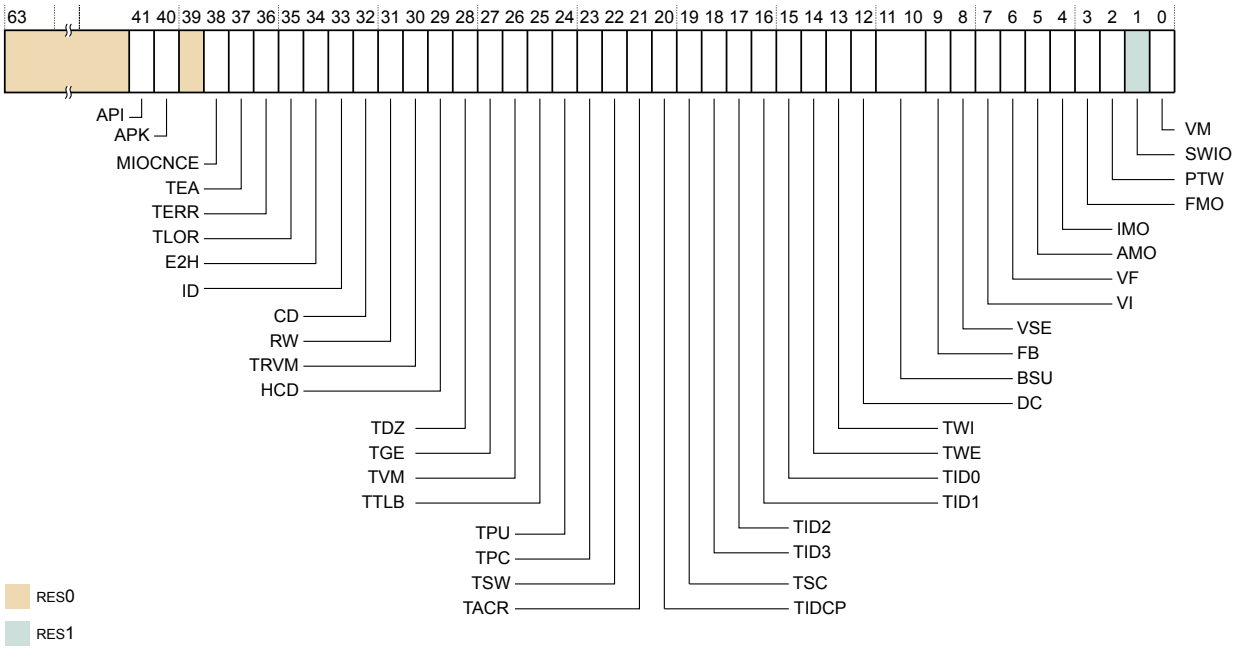

Figure B2-60 HCR_EL2 bit assignments

RES0, [63:42]

RES0 Reserved

API, [41]

Controls the use of instructions related to Pointer Authentication. The possible values are:

0 Use of these instructions in Non-secure EL0 when HCR_EL2.TGE==0 || HCR_EL2.E2H==0, or in Non-secure EL1 when the instructions are enabled for the Non-secure EL1 translation regime is trapped to EL2. The ESR_EL2.EC code in the event of such traps is 0x9, and the ESR_EL2.ISS field is RES0.

1 This control does not cause any instructions to be trapped.

This field resets to an architecturally UNKNOWN value.

APK, [40]

The possible values are:

0 Access to the authentication key registers from non-secure EL1 are trapped to EL2. The ESR_EL2.EC code in the event of such traps is 0x18, and the ESR_EL2.ISS field takes the standard meanings.

1 Access to these registers are not trapped to EL2 by this mechanism. If EL2 is not implemented, the system behaves as if HCR_EL2.APK = 1.

This field resets to an architecturally UNKNOWN value.

RES0, [39]

RES0 Reserved

MIOCNCE, [38]

Mismatched Inner/Outer Cacheable Non-Coherency Enable, for the Non-secure EL1 and EL0 translation regime.

RW, [31]

RES1 Reserved

HCD, [29]

RES0 Reserved

TGE, [27]

Traps general exceptions. If this bit is set, and SCR_EL3.NS is set, then:

- All exceptions that would be routed to EL1 are routed to EL2.
- The SCTLR_EL1.M bit is treated as `0` regardless of its actual state, other than for reading the bit.
- The HCR_EL2.FMO, IMO, and AMO bits are treated as `1` regardless of their actual state, other than for reading the bits.
- 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.

HCR_EL2.TGE must not be cached in a TLB.

When the value of SCR_EL3.NS is 0 the core behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

TID3, [18]

Traps ID group 3 registers. The possible values are:

`0` ID group 3 register accesses are not trapped.

`1` Reads to ID group 3 registers executed from Non-secure EL1 are trapped to EL2.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for the registers covered by this setting.

Configurations

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

RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.75 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

The core does not use this register, ID_AA64AFR0_EL1 is RES0.

B2.76 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

The core does not use this register, ID_AA64AFR0_EL1 is RES0.

B2.77 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1

Provides top-level information about the debug system in AArch64.

Bit field descriptions

ID_AA64DFR0_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

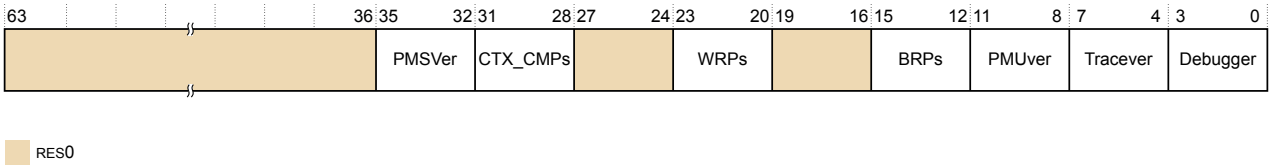

Figure B2-61 ID_AA64DFR0_EL1 bit assignments

RES0, [63:36]

RES0 Reserved

PMSVer, [35:32]

Statistical Profiling Extension version:

0x1 Version 1 of the Statistical Profiling extension is present.

CTX_CMPs, [31:28]

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints:

0x1 Two breakpoints are context-aware.

RES0, [27:24]

RES0 Reserved

WRPs, [23:20]

The number of watchpoints minus 1:

0x3 Four watchpoints

RES0, [19:16]

RES0 Reserved

BRPs, [15:12]

The number of breakpoints minus 1:

0x5 Six breakpoints

PMUVer, [11:8]

Performance Monitors Extension version:

0x4 Performance monitor system registers are implemented, PMUv3.

TraceVer, [7:4]

Trace extension:

0x0 Trace system registers are not implemented.

DebugVer, [3:0]

Debug architecture version:

`0x8` Armv8-A debug architecture is implemented.

Configurations

ID_AA64DFR0_EL1 is architecturally mapped to external register EDDFR.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.78 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1

This register is reserved for future expansion of top level information about the debug system in AArch64 state.

B2.79 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1

The ID_AA64ISAR0_EL1 provides information about the instructions implemented in AArch64 state, including the instructions that are provided by the Cryptographic Extension.

Bit field descriptions

ID_AA64ISAR0_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

The optional Cryptographic Extension is not included in the base product of the core. Arm requires licensees to have contractual rights to obtain the Cryptographic Extension.

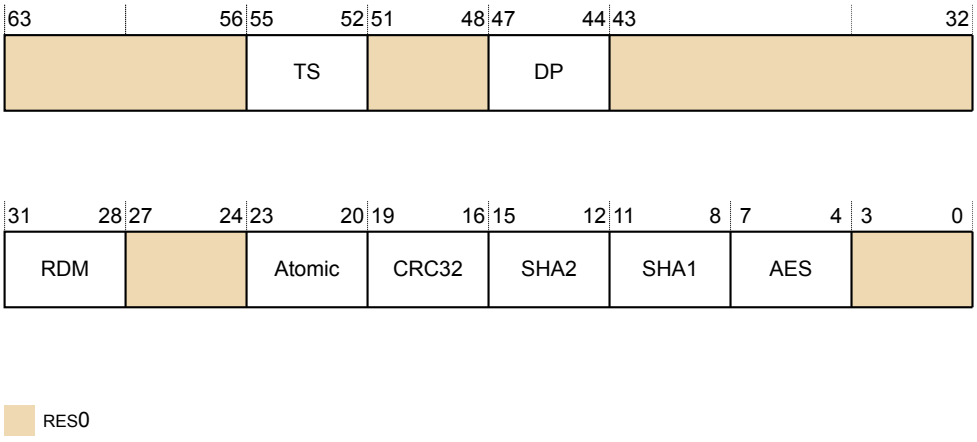

Figure B2-62 ID_AA64ISAR0_EL1 bit assignments

RES0, [63:56]

RES0 Reserved

TS, [55:52]

Indicates support for flag manipulation instructions.

0x1 CFINV, RMIF, SETF16, and SETF8 instructions are implemented.

RES0, [51:48]

RES0 Reserved

DP, [47:44]

Indicates whether Dot Product support instructions are implemented.

0x1 UDOT, SDOT instructions are implemented.

RES0, [43:32]

RES0 Reserved

RDM, [31:28]

Indicates whether SQRDMLAH and SQRDMLSH instructions in AArch64 are implemented.

0x1 SQRDMLAH and SQRDMLSH instructions are implemented.

RES0, [27:24]

RES0 Reserved

Atomic, [23:20]

Indicates whether Atomic instructions in AArch64 are implemented. The value is:

0x2 LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions are implemented.

.

CRC32, [19:16]

Indicates whether CRC32 instructions are implemented. The value is:

0x1 CRC32 instructions are implemented.

SHA2, [15:12]

Indicates whether SHA2 instructions are implemented. The possible values are:

0x0 No SHA2 instructions are implemented. This is the value if the core implementation does not include the Cryptographic Extension.

0x1 SHA256H, SHA256H2, SHA256U0, and SHA256U1 implemented. This is the value if the core implementation includes the Cryptographic Extension.

SHA1, [11:8]

Indicates whether SHA1 instructions are implemented. The possible values are:

0x0 No SHA1 instructions implemented. This is the value if the core implementation does not include the Cryptographic Extension.

0x1 SHA1C, SHA1P, SHA1M, SHA1SU0, and SHA1SU1 implemented. This is the value if the core implementation includes the Cryptographic Extension.

AES, [7:4]

Indicates whether AES instructions are implemented. The possible values are:

0x0 No AES instructions implemented. This is the value if the core implementation does not include the Cryptographic Extension.

0x2 AESE, AESD, AESMC, and AESIMC implemented, plus PMULL and PMULL2 instructions operating on 64-bit data. This is the value if the core implementation includes the Cryptographic Extension.

RES0, [3:0]

RES0 Reserved

Configurations

ID_AA64ISAR0_EL1 is architecturally mapped to external register ID_AA64ISAR0.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.80 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1

The ID_AA64ISAR1_EL1 provides information about the instructions implemented in AArch64 state.

Bit field descriptions

ID_AA64ISAR1_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

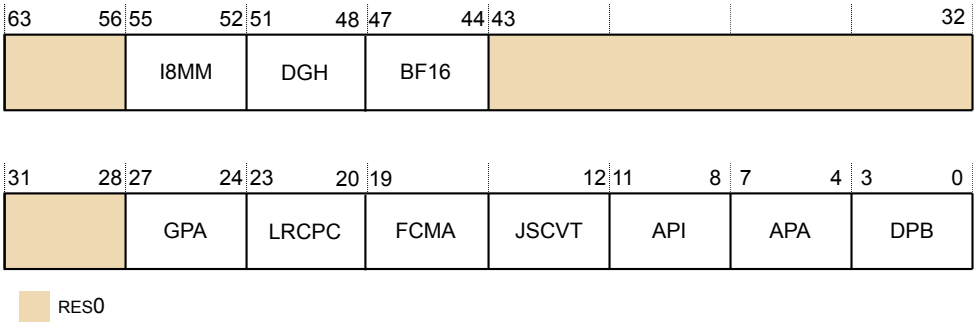

Figure B2-63 ID_AA64ISAR1_EL1 bit assignments

RES0, [63:56]

RES0 Reserved

RES0, [43:32]

RES0 Reserved

RES0, [31:28]

RES0 Reserved

GPA, [27:24]

Indicates whether QARMA or Architected algorithm is implemented in the PE for generic code authentication, in AArch64 state.

0x1 Generic Authentication using the QARMA algorithm is implemented. This involves the PACGA instruction.

LRCPC, [23:20]

Indicates whether load-acquire (LDA) instructions are implemented for a Release Consistent core consistent RCPC model.

0x2 The LDAPUR*, STLUR*, and LDAPR* instructions are implemented.

FCMA, [19:16]

Indicates support for complex number addition and multiplication, where numbers are stored in vectors.

0x0 The FCMLA and FCADD instructions are not implemented.

JSCVT, [15:12]

Indicates support for javascript conversion from double-precision floating point values to integers in AArch64 state.

0x0 The FJCVTZS instruction is not implemented.

API, [11:8]

Indicates whether an IMPLEMENTATION DEFINED algorithm is implemented in the PE for address authentication, in AArch64 state.

`0x0` Address Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.

APA, [7:4]

Indicates whether QARMA or Architected algorithm is implemented in the PE for address authentication, in AArch64 state.

`0x3` Address Authentication using the QARMA algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning FALSE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE. This applies to all Pointer Authentication instructions other than the PACGA instruction.

DPB, [3:0]

Data Persistence write-back. Indicates support for the DC CVAP and DC CVADP instructions in AArch64 state.

`0x1` `DC CVAP` supported in AArch64

All other values are reserved.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.81 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1

The ID_AA64MMFR0_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

Bit field descriptions

ID_AA64MMFR0_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

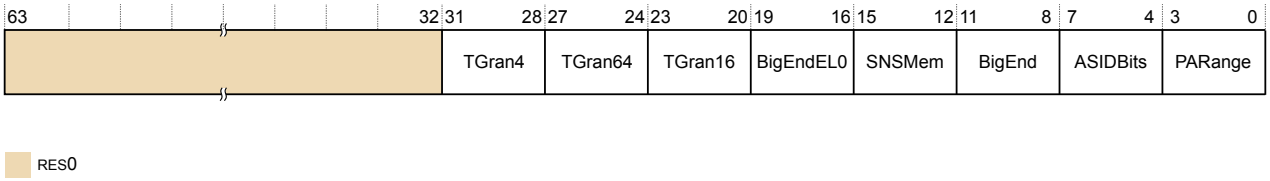

Figure B2-64 ID_AA64MMFR0_EL1 bit assignments

RES0, [63:32]

RES0 Reserved

TGran4, [31:28]

Support for 4KB memory translation granule size:

0x0 4KB granule supported

TGran64, [27:24]

Support for 64KB memory translation granule size:

0x0 64KB granule supported

TGran16, [23:20]

Support for 16KB memory translation granule size:

0x1 16KB granule supported

BigEndEL0, [19:16]

Mixed-endian support only at EL0:

0x0 No mixed-endian support at EL0. The SCTLR_EL1.E0E bit has a fixed value.

SNSMem, [15:12]

Secure versus Non-secure Memory distinction:

0x1 Supports a distinction between Secure and Non-secure Memory

BigEnd, [11:8]

Mixed-endian configuration support:

0x1 Mixed-endian support. The SCTLR_ELx.EE and SCTLR_EL1.E0E bits can be configured.

ASIDBits, [7:4]

Number of ASID bits:

0x2 16 bits

PARange, [3:0]

Physical address range supported:

`0x2` 40 bits, 1TB

The supported Physical Address Range is 40-bits. Other cores in the DSU may support a different Physical Address Range.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.82 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1

The ID_AA64MMFR1_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

Bit field descriptions

ID_AA64MMFR1_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

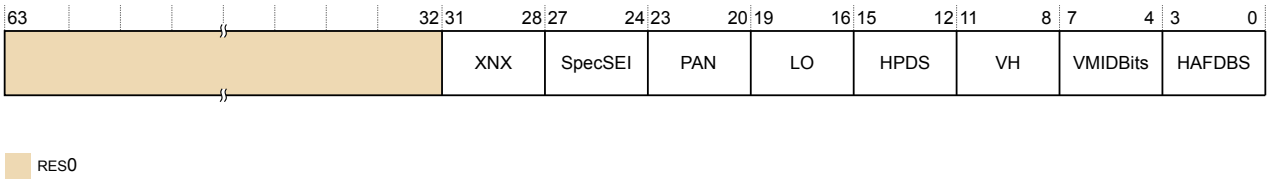

Figure B2-65 ID_AA64MMFR1_EL1 bit assignments

RES0, [63:32]

RES0 Reserved

XNX, [31:28]

Indicates whether provision of EL0 vs EL1 execute-never control at Stage 2 is supported.

0x1 EL0/EL1 execute control distinction at Stage 2 bit is supported. All other values are reserved.

SpecSEI, [27:24]

Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches.

0x0 The PE never generates an SError interrupt due to an External abort on a speculative read.

PAN, [23:20]

Privileged Access Never. Indicates support for the PAN bit in PSTATE, SPSR_EL1, SPSR_EL2, SPSR_EL3, and DSPSR_EL0.

0x2 PAN supported and AT S1E1RP and AT S1E1WP instructions supported.

LO, [19:16]

Indicates support for LORegions.

0x1 LORegions supported

HPDS, [15:12]

Presence of Hierarchical Disables. Enables an operating system or hypervisor to hand over up to 4 bits of the last level page table descriptor (bits[62:59] of the page table entry) for use by hardware for IMPLEMENTATION DEFINED usage. The value is:

0x2 Hierarchical Permission Disables and Hardware allocation of bits[62:59] supported

VH, [11:8]

Indicates whether Virtualization Host Extensions supported.

`0x1` Virtualization Host Extensions supported

VMIDBits, [7:4]

Indicates the number of VMID bits supported.

`0x2` 16 bits supported

HAFDBS, [3:0]

Indicates the support for hardware updates to Access flag and dirty state in translation tables.

`0x2` Hardware update of both the Access flag and dirty state supported in hardware

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.83 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1

The ID_AA64MMFR2_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

Bit field descriptions

ID_AA64MMFR2_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

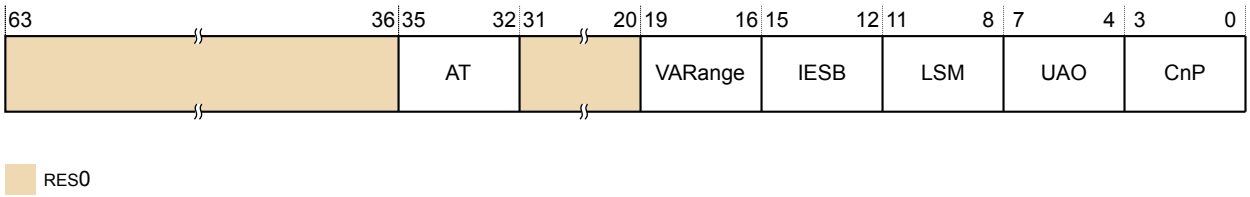

Figure B2-66 ID_AA64MMFR2_EL1 bit assignments

RES0, [63:36]

RES0 Reserved

AT, [35:32]

Identifies support for unaligned single-copy atomicity and atomic functions. The value is:

0x1 Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.

RES0, [31:20]

RES0 Reserved

VARange, [19:16]

Indicates support for a larger virtual address. The value is:

0x0 VMSAv8-64 supports 48-bit virtual addresses.

IESB, [15:12]

Indicates whether an implicit Error Synchronization Barrier has been inserted. The value is:

0x1 SCTLR_ELx.IESB implicit `ErrorSynchronizationBarrier` control implemented.

LSM, [11:8]

Indicates whether LDM and STM ordering control bits are supported. The value is:

0x0 LSMAOE and nTLSMD bit not supported.

UAO, [7:4]

Indicates the presence of the *User Access Override* (UAO). The value is:

0x1 UAO is supported.

CnP, [3:0]

Common not Private. Indicates whether a TLB entry is pointed at a translation table base register that is a member of a common set. The value is:

0x1 CnP bit is supported.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.84 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1

The ID_AA64PFR0_EL1 provides additional information about implemented core features in AArch64.

The optional Advanced SIMD and floating-point support is not included in the base product of the core. Arm requires licensees to have contractual rights to obtain the Advanced SIMD and floating-point support.

Bit field descriptions

ID_AA64PFR0_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

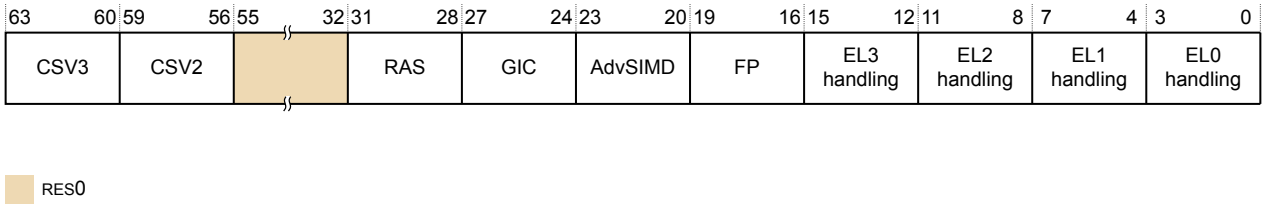

Figure B2-67 ID_AA64PFR0_EL1 bit assignments

CSV3, [63:60]

`0x1` Data loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes to be used by instructions newer than the load in the speculative sequence. This is the reset value.

All other values reserved

CSV2, [59:56]

`0x1` Branch targets trained in one context cannot affect speculative execution in a different hardware described context. This is the reset value.

All other values reserved

RES0, [55:32]

RES0 Reserved

RAS, [31:28]

RAS extension version. The possible values are:

`0x0` RAS extension is not present. This is the value if the core implementation does not have ECC present.

`0x1` Version 1 of the RAS extension is present. This is the value if the core implementation has ECC present.

GIC, [27:24]

GIC CPU interface:

`0x0` GIC CPU interface is disabled, GICCDISABLE is HIGH, or not implemented.

`0x1` GIC CPU interface is implemented and enabled, GICCDISABLE is LOW.

AdvSIMD, [23:20]

Advanced SIMD. The possible values are:

`0x1` Advanced SIMD, including half-precision support, is implemented.

FP, [19:16]

Floating-point. The possible values are:

`0x1` Floating-point, including half-precision support, is implemented.

EL3 handling, [15:12]

EL3 exception handling:

`0x1` Instructions can be executed at EL3 in AArch64 state only.

EL2 handling, [11:8]

EL2 exception handling:

`0x1` Instructions can be executed at EL3 in AArch64 state only.

EL1 handling, [7:4]

EL1 exception handling. The possible values are:

`0x1` Instructions can be executed at EL3 in AArch64 state only.

EL0 handling, [3:0]

EL0 exception handling. The possible values are:

`0x2` Instructions can be executed at EL0 in AArch64 or AArch32 state.

Configurations

ID_AA64PFR0_EL1 is architecturally mapped to External register EDPFR.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.85 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1

The ID_AA64PFR1_EL1 provides additional information about implemented core features in AArch64.

Bit field descriptions

ID_AA64PFR1_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is read-only.

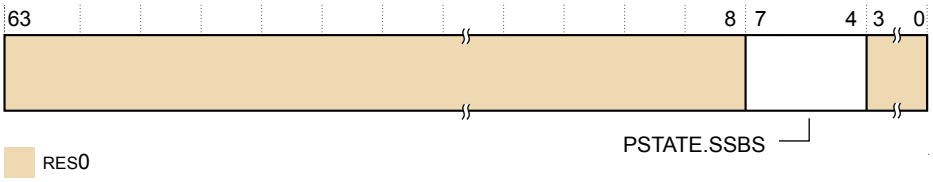

Figure B2-68 ID_AA64PFR1_EL1 bit assignments

RES0, [63:8]

RES0 Reserved

SSBS, [7:4]

AArch64 provides the PSTATE.SSBS mechanism to mark regions that are *Speculative Store Bypassing Safe* (SSBS).

`0x01` AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe, but does not implement the MSR/MRS instructions to directly read and write the PSTATE.SSBS field.

RES0, [3:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.86 ID_AFR0_EL1, AArch32 Auxiliary Feature Register 0, EL1

The ID_AFR0_EL1 provides information about the IMPLEMENTATION DEFINED features of the PE in AArch32. This register is not used in the Cortex-A78C core.

Bit field descriptions

ID_AFR0_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

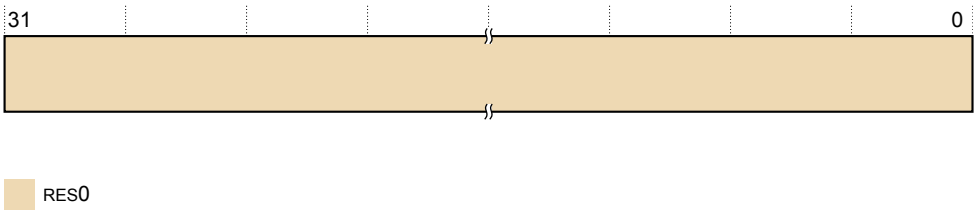

Figure B2-69 ID_AFR0_EL1 bit assignments

RES0, [31:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.87 ID_DFR0_EL1, AArch32 Debug Feature Register 0, EL1

The ID_DFR0_EL1 provides top-level information about the debug system in AArch32.

Bit field descriptions

ID_DFR0_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

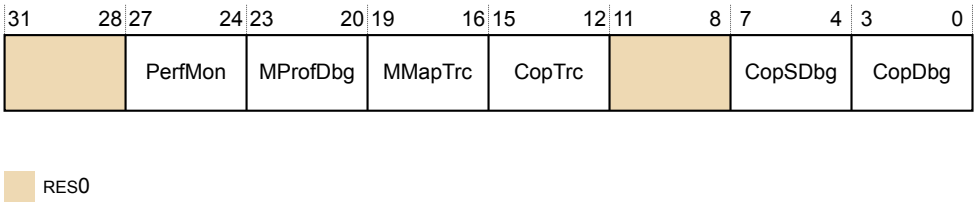

Figure B2-70 ID_DFR0_EL1 bit assignments

RES0, [31:28]

RES0 Reserved

PerfMon, [27:24]

Indicates support for performance monitor model:

4 Support for *Performance Monitoring Unit version 3* (PMUv3) System registers, with a 16-bit evtCount field

MProfDbg, [23:20]

Indicates support for memory-mapped debug model for M profile cores:

0 This product does not support M profile Debug architecture.

MMapTrc, [19:16]

Indicates support for memory-mapped trace model:

1 Support for Arm trace architecture, with memory-mapped access

In the Trace registers, the ETMIDR gives more information about the implementation.

CopTrc, [15:12]

Indicates support for coprocessor-based trace model:

0 This product does not support Arm trace architecture.

RES0, [11:8]

RES0 Reserved

CopSDbg, [7:4]

Indicates support for coprocessor-based Secure debug model:

8 This product supports the Armv8.2 Debug architecture.

CopDbg, [3:0]

Indicates support for coprocessor-based debug model:

8 This product supports the Armv8.2 Debug architecture.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1

The ID_ISAR0_EL1 provides information about the instruction sets implemented by the core in AArch32.

Bit field descriptions

ID_ISAR0_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

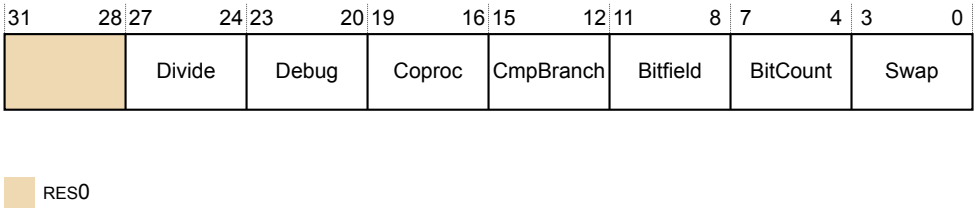

Figure B2-71 ID_ISAR0_EL1 bit assignments

RES0, [31:28]

RES0 Reserved

Divide, [27:24]

Indicates the implemented Divide instructions:

0x2
- SDIV and UDIV in the T32 instruction set
- SDIV and UDIV in the A32 instruction set

Debug, [23:20]

Indicates the implemented Debug instructions:

0x1 BKPT

Coproc, [19:16]

Indicates the implemented Coprocessor instructions:

0x0 None implemented, except for instructions separately attributed by the architecture to provide access to AArch32 System registers and System instructions

CmpBranch, [15:12]

Indicates the implemented combined Compare and Branch instructions in the T32 instruction set:

0x1 CBNZ and CBZ

Bitfield, [11:8]

Indicates the implemented bit field instructions:

0x1 BFC, BFI, SBFX, and UBFX

BitCount, [7:4]

Indicates the implemented Bit Counting instructions:

0x1 CLZ

Swap, [3:0]

Indicates the implemented Swap instructions in the A32 instruction set:

`0x0` None implemented

Configurations

In an AArch64-only implementation, this register is UNKNOWN.

Must be interpreted with ID_ISAR1_EL1, ID_ISAR2_EL1, ID_ISAR3_EL1, ID_ISAR4_EL1, ID_ISAR5_EL1, and ID_ISAR6_EL1. See:

- *B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
- *B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
- *B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
- *B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287
- *B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289
- *B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1

The ID_ISAR1_EL1 provides information about the instruction sets implemented by the core in AArch32.

Bit field descriptions

ID_ISAR1_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

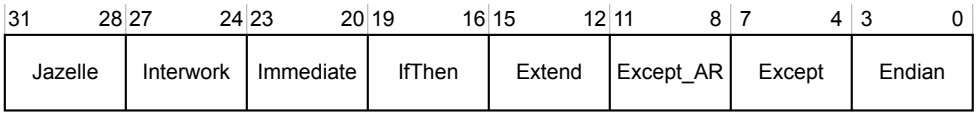

Figure B2-72 ID_ISAR1_EL1 bit assignments

Jazelle, [31:28]

Indicates the implemented Jazelle state instructions:

`0x1` Adds the `BXJ` instruction, and the J bit in the PSR

Interwork, [27:24]

Indicates the implemented Interworking instructions:

`0x3`
- The `BX` instruction, and the T bit in the PSR
- The `BLX` instruction. The PC loads have `BX`-like behavior.
- Data-processing instructions in the A32 instruction set with the PC as the destination and the S bit clear, have `BX`-like behavior.

Immediate, [23:20]

Indicates the implemented data-processing instructions with long immediates:

`0x1`
- The `MOVT` instruction
- The `MOV` instruction encodings with zero-extended 16-bit immediates
- The T32 `ADD` and `SUB` instruction encodings with zero-extended 12-bit immediates, and other `ADD`, `ADR`, and `SUB` encodings cross-referenced by the pseudocode for those encodings

IfThen, [19:16]

Indicates the implemented If-Then instructions in the T32 instruction set:

`0x1` The `IT` instructions, and the IT bits in the PSRs

Extend, [15:12]

Indicates the implemented Extend instructions:

`0x2`
- The `SXTB`, `SXTH`, `UXTB`, and `UXTH` instructions
- The `SXTB16`, `SXTAB`, `SXTAB16`, `SXTAH`, `UXTB16`, `UXTAB`, `UXTAB16`, and `UXTAH` instructions

Except_AR, [11:8]

Indicates the implemented A profile exception-handling instructions:

`0x1` The `SRS` and `RFE` instructions, and the A profile forms of the `CPS` instruction

Except, [7:4]

Indicates the implemented exception-handling instructions in the A32 instruction set:

`0x1` The `LDM` (exception return), `LDM` (user registers), and `STM` (user registers) instruction versions

Endian, [3:0]

Indicates the implemented Endian instructions:

`0x1` The `SETEND` instruction, and the E bit in the PSRs

Configurations

In an AArch64-only implementation, this register is UNKNOWN.

Must be interpreted with ID_ISAR0_EL1, ID_ISAR2_EL1, ID_ISAR3_EL1, ID_ISAR4_EL1, ID_ISAR5_EL1, and ID_ISAR6_EL1. See:

- *B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
- *B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
- *B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
- *B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287
- *B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289
- *B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1

The ID_ISAR2_EL1 provides information about the instruction sets implemented by the core in AArch32.

Bit field descriptions

ID_ISAR2_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

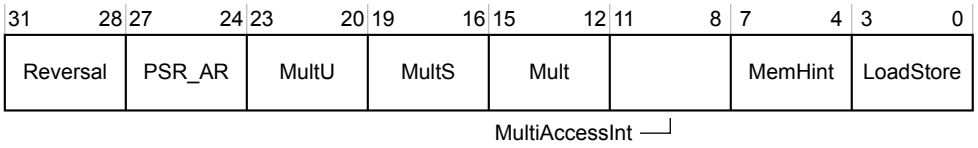

Figure B2-73 ID_ISAR2_EL1 bit assignments

Reversal, [31:28]

Indicates the implemented Reversal instructions:

`0x2` The `REV`, `REV16`, `REVSH`, and `RBIT` instructions

PSR_AR, [27:24]

Indicates the implemented A and R profile instructions to manipulate the PSR:

`0x1` The `MRS` and `MSR` instructions, and the exception return forms of data-processing instructions

The exception return forms of the data-processing instructions are:

- In the A32 instruction set, data-processing instructions with the PC as the destination and the S bit set
- In the T32 instruction set, the `SUBSPC`, `LR`, `#N` instruction

MultU, [23:20]

Indicates the implemented advanced unsigned Multiply instructions:

`0x2` The `UMULL`, `UMLAL`, and `UMAAL` instructions

MultS, [19:16]

Indicates the implemented advanced signed Multiply instructions.

`0x3`
- The `SMULL` and `SMLAL` instructions
- The `SMLABB`, `SMLABT`, `SMLALBB`, `SMLALBT`, `SMLALTB`, `SMLALTT`, `SMLATB`, `SMLATT`, `SMLAWB`, `SMLAWT`, `SMULBB`, `SMULBT`, `SMULTB`, `SMULTT`, `SMULWB`, `SMULWT` instructions, and the Q bit in the PSRs
- The `SMLAD`, `SMLADX`, `SMLALD`, `SMLALDX`, `SMLSD`, `SMLSDX`, `SMLSLD`, `SMLSLDX`, `SMMLA`, `SMMLAR`, `SMMLS`, `SMMLSR`, `SMMUL`, `SMMULR`, `SMUAD`, `SMUADX`, `SMUSD`, and `SMUSDX` instructions

Mult, [15:12]

Indicates the implemented additional Multiply instructions:

`0x2` The `MUL`, `MLA` and `MLS` instructions

MultiAccessInt, [11:8]

Indicates the support for interruptible multi-access instructions:

`0x0` No support. This means the LDM and STM instructions are not interruptible.

MemHint, [7:4]

Indicates the implemented memory hint instructions:

`0x4` The `PLD`, `PLI`, and `PLDW`instructions

LoadStore, [3:0]

Indicates the implemented additional load/store instructions:

`0x2` The `LDRD` and `STRD` instructions

The Load Acquire (`LDAB`, `LDAH`, `LDA`, `LDAEXB`, `LDAEXH`, `LDAEX`, and `LDAEXD`) and Store Release (`STLB`, `STLH`, `STL`, `STLEXB`, `STLEXH`, `STLEX`, and `STLEXD`) instructions.

Configurations

In an AArch64-only implementation, this register is UNKNOWN.

Must be interpreted with ID_ISAR0_EL1, ID_ISAR1_EL1, ID_ISAR3_EL1, ID_ISAR4_EL1, ID_ISAR5_EL1, and ID_ISAR6_EL1. See:

- *B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
- *B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
- *B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
- *B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287.
- *B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289
- *B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1

The ID_ISAR3_EL1 provides information about the instruction sets implemented by the core in AArch32.

Bit field descriptions

ID_ISAR3_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

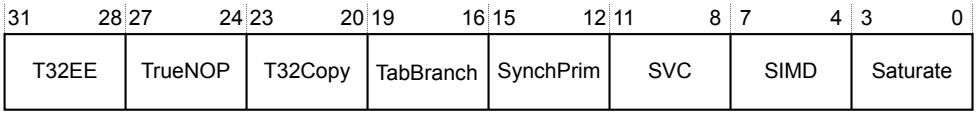

Figure B2-74 ID_ISAR3_EL1 bit assignments

T32EE, [31:28]

Indicates the implemented T32EE instructions:

`0x0` None implemented

TrueNOP, [27:24]

Indicates support for True NOP instructions:

`0x1` True `NOP` instructions in both the A32 and T32 instruction sets, and additional NOP-compatible hints

T32Copy, [23:20]

Indicates the support for T32 non flag-setting `MOV` instructions:

`0x1` Support for T32 instruction set encoding T1 of the `MOV` (register) instruction, copying from a low register to a low register

TabBranch, [19:16]

Indicates the implemented Table Branch instructions in the T32 instruction set.

`0x1` The `TBB` and `TBH` instructions

SynchPrim, [15:12]

Indicates the implemented Synchronization Primitive instructions:

`0x2`
- The `LDREX` and `STREX` instructions
- The `CLREX`, `LDREXB`, `STREXB`, and `STREXH` instructions
- The `LDREXD` and `STREXD` instructions

SVC, [11:8]

Indicates the implemented SVC instructions:

`0x1` The `SVC` instruction

SIMD, [7:4]

Indicates the implemented *Single Instruction Multiple Data* (SIMD) instructions.

`0x3` • The `SSAT` and `USAT` instructions, and the Q bit in the PSRs
• The `PKHBT`, `PKHTB`, `QADD16`, `QADD8`, `QASX`, `QSUB16`, `QSUB8`, `QSAX`, `SADD16`, `SADD8`, `SASX`, `SEL`, `SHADD16`, `SHADD8`, `SHASX`, `SHSUB16`, `SHSUB8`, `SHSAX`, `SSAT16`, `SSUB16`, `SSUB8`, `SSAX`, `SXTAB16`, `SXTB16`, `UADD16`, `UADD8`, `UASX`, `UHADD16`, `UHADD8`, `UHASX`, `UHSUB16`, `UHSUB8`, `UHSAX`, `UQADD16`, `UQADD8`, `UQASX`, `UQSUB16`, `UQSUB8`, `UQSAX`, `USAD8`, `USADA8`, `USAT16`, `USUB16`, `USUB8`, `USAX`, `UXTAB16`, `UXTB16` instructions, and the GE[3:0] bits in the PSRs

The SIMD field relates only to implemented instructions that perform SIMD operations on the general-purpose registers. In an implementation that supports Advanced SIMD and floating-point instructions, MVFR0, MVFR1, and MVFR2 give information about the implemented Advanced SIMD instructions.

Saturate, [3:0]

Indicates the implemented Saturate instructions:

`0x1` The `QADD`, `QDADD`, `QDSUB`, `QSUB` Q bit in the PSRs

Configurations

In an AArch64-only implementation, this register is UNKNOWN.

Must be interpreted with ID_ISAR0_EL1, ID_ISAR1_EL1, ID_ISAR2_EL1, ID_ISAR4_EL1, ID_ISAR5_EL1, and ID_ISAR6_EL1. See:
- *B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
- *B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
- *B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
- *B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287
- *B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289
- *B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1

The ID_ISAR4_EL1 provides information about the instruction sets implemented by the core in AArch32.

Bit field descriptions

ID_ISAR4_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

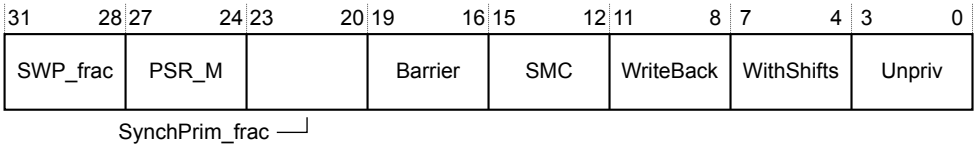

Figure B2-75 ID_ISAR4_EL1 bit assignments

SWP_frac, [31:28]

Indicates support for the memory system locking the bus for `SWP` or `SWPB` instructions:

`0x0` `SWP` and `SWPB` instructions not implemented

PSR_M, [27:24]

Indicates the implemented M profile instructions to modify the PSRs:

`0x0` None implemented

SynchPrim_frac, [23:20]

This field is used with the ID_ISAR3.SynchPrim field to indicate the implemented Synchronization Primitive instructions:

`0x0`
- The `LDREX` and `STREX` instructions
- The `CLREX`, `LDREXB`, `LDREXH`, `STREXB`, and `STREXH` instructions
- The `LDREXD` and `STREXD` instructions

Barrier, [19:16]

Indicates the supported Barrier instructions in the A32 and T32 instruction sets:

`0x1` The `DMB`, `DSB`, and `ISB` barrier instructions

SMC, [15:12]

Indicates the implemented `SMC` instructions:

`0x0` None implemented

WriteBack, [11:8]

Indicates the support for Write-Back addressing modes:

`0x1` Core supports all the Write-Back addressing modes as defined in Armv8-A

WithShifts, [7:4]

Indicates the support for instructions with shifts:

`0x4`
- Support for shifts of loads and stores over the range LSL 0-3
- Support for other constant shift options, both on load/store and other instructions
- Support for register-controlled shift options

Unpriv, [3:0]

Indicates the implemented unprivileged instructions:

`0x2`
- The `LDRBT`, `LDRT`, `STRBT`, and `STRT` instructions
- The `LDRHT`, `LDRSBT`, `LDRSHT`, and `STRHT` instructions

Configurations

In an AArch64-only implementation, this register is UNKNOWN.

Must be interpreted with ID_ISAR0_EL1, ID_ISAR1_EL1, ID_ISAR2_EL1, ID_ISAR3_EL1, ID_ISAR5_EL1, and ID_ISAR6_EL1. See:
- *B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
- *B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
- *B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
- *B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
- *B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289
- *B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1

The ID_ISAR5_EL1 provides information about the instruction sets that the core implements.

Bit field descriptions

ID_ISAR5_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

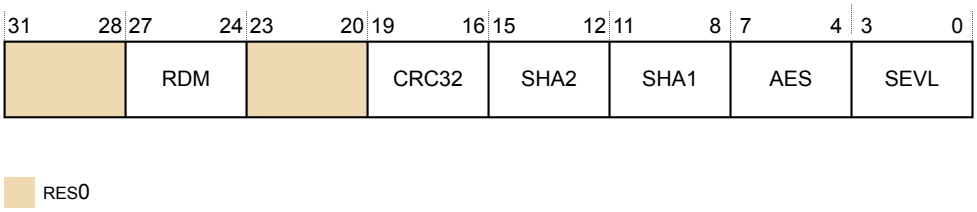

Figure B2-76 ID_ISAR5_EL1 bit assignments

RES0, [31:28]

RES0 Reserved

RDM, [27:24]

VQRDMLAH and VQRDMLSH instructions in AArch32. The value is:

`0x1` VQRDMLAH and VQRDMLSH instructions are implemented.

RES0, [23:20]

RES0 Reserved

CRC32, [19:16]

Indicates whether CRC32 instructions are implemented in AArch32 state. The value is:

`0x1` `CRC32B`, `CRC32H`, `CRC32W`, `CRC32CB`, `CRC32CH`, and `CRC32CW` instructions are implemented.

SHA2, [15:12]

Indicates whether SHA2 instructions are implemented in AArch32 state. The possible values are:

`0x0` No SHA2 instructions implemented. This is the value when the Cryptographic Extensions are not implemented or are disabled.

`0x1` `SHA256H`, `SHA256H2`, `SHA256SU0`, and `SHA256SU1` instructions are implemented. This is the value when the Cryptographic Extensions are implemented and enabled.

SHA1, [11:8]

Indicates whether SHA1 instructions are implemented in AArch32 state. The possible values are:

`0x0` No SHA1 instructions implemented. This is the value when the Cryptographic Extensions are not implemented or are disabled.

`0x1` `SHA1C`, `SHA1P`, `SHA1M`, `SHA1H`, `SHA1SU0`, and `SHA1SU1` instructions are implemented. This is the value when the Cryptographic Extensions are implemented and enabled.

AES, [7:4]

Indicates whether AES instructions are implemented in AArch32 state. The possible values are:

`0x0` No AES instructions implemented. This is the value when the Cryptographic Extensions are not implemented or are disabled.

`0x2`
- `AESE`, `AESD`, `AESMC`, and `AESIMC` implemented
- `PMULL` and `PMULL2` instructions operating on 64-bit data

This is the value when the Cryptographic Extensions are implemented and enabled.

SEVL, [3:0]

Indicates whether the `SEVL` instruction is implemented:

`0x1` `SEVL` implemented to send event local

Configurations

ID_ISAR5 must be interpreted with ID_ISAR0_EL1, ID_ISAR1_EL1, ID_ISAR2_EL1, ID_ISAR3_EL1, ID_ISAR4_EL1, and ID_ISAR6_EL1. See:

- *B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
- *B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
- *B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
- *B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
- *B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287
- *B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1* on page B2-291

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.94 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6, EL1

The ID_ISAR6_EL1 provides information about the instruction sets that the core implements.

Bit field descriptions

ID_ISAR6_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

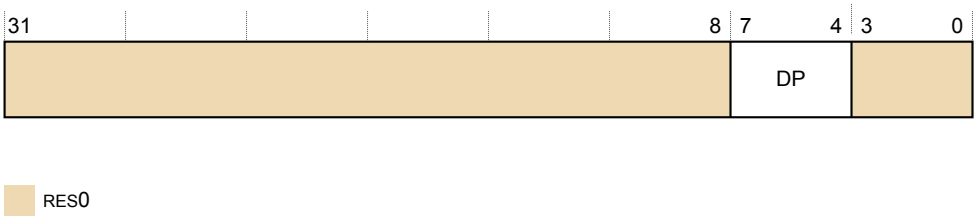

Figure B2-77 ID_ISAR6_EL1 bit assignments

RES0, [31:8]

RES0 Reserved

DP, [7:4]

UDOT and SDOT instructions. The value is:

`0b0001` UDOT and SDOT instructions are implemented.

RES0, [3:0]

RES0 Reserved

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

ID_ISAR6_EL1 must be interpreted with ID_ISAR0_EL1, ID_ISAR1_EL1, ID_ISAR2_EL1, ID_ISAR3_EL1, ID_ISAR4_EL1, and ID_ISAR5_EL1. See:

- *B2.88 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0, EL1* on page B2-279
- *B2.89 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1, EL1* on page B2-281
- *B2.90 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2, EL1* on page B2-283
- *B2.91 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3, EL1* on page B2-285
- *B2.92 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4, EL1* on page B2-287
- *B2.93 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1* on page B2-289

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.95 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0, EL1

The ID_MMFR0_EL1 provides information about the memory model and memory management support in AArch32.

Bit field descriptions

ID_MMFR0_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

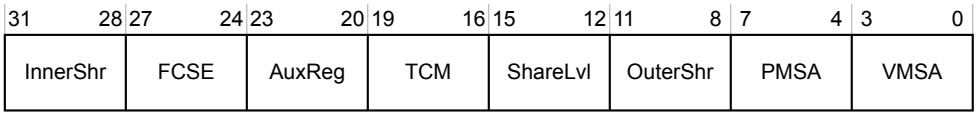

Figure B2-78 ID_MMFR0_EL1 bit assignments

InnerShr, [31:28]

Indicates the innermost shareability domain implemented:

`0x1` Implemented with hardware coherency support

FCSE, [27:24]

Indicates support for *Fast Context Switch Extension* (FCSE):

`0x0` Not supported

AuxReg, [23:20]

Indicates support for Auxiliary registers:

`0x2` Support for Auxiliary Fault Status Registers (AIFSR and ADFSR) and Auxiliary Control Register

TCM, [19:16]

Indicates support for TCMs and associated DMAs:

`0x0` Not supported

ShareLvl, [15:12]

Indicates the number of shareability levels implemented:

`0x1` Two levels of shareability implemented

OuterShr, [11:8]

Indicates the outermost shareability domain implemented:

`0x1` Implemented with hardware coherency support

PMSA, [7:4]

Indicates support for a *Protected Memory System Architecture* (PMSA):

`0x0` Not supported

VMSA, [3:0]

Indicates support for a *Virtual Memory System Architecture* (VMSA).

`0x5` Support for:

- VMSAv7, with support for remapping and the Access flag
- The PXN bit in the Short-descriptor translation table format descriptors
- The Long-descriptor translation table format

Configurations

Must be interpreted with ID_MMFR1_EL1, ID_MMFR2_EL1, ID_MMFR3_EL1, and ID_MMFR4_EL1. See:

- *B2.96 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1, EL1* on page B2-294
- *B2.97 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2, EL1* on page B2-296
- *B2.98 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3, EL1* on page B2-298
- *B2.99 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4, EL1* on page B2-300

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.96 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1, EL1

The ID_MMFR1_EL1 provides information about the memory model and memory management support in AArch32.

Bit field descriptions

ID_MMFR1_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

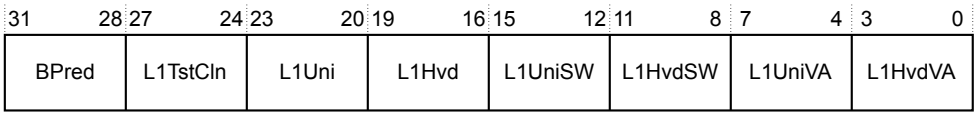

Figure B2-79 ID_MMFR1_EL1 bit assignments

BPred, [31:28]

Indicates branch predictor management requirements:

`0x4` For execution correctness, branch predictor requires no flushing at any time.

L1TstCln, [27:24]

Indicates the supported L1 Data cache test and clean operations, for Harvard or unified cache implementation:

`0x0` None supported

L1Uni, [23:20]

Indicates the supported entire L1 cache maintenance operations, for a unified cache implementation:

`0x0` None supported

L1Hvd, [19:16]

Indicates the supported entire L1 cache maintenance operations, for a Harvard cache implementation:

`0x0` None supported

L1UniSW, [15:12]

Indicates the supported L1 cache line maintenance operations by set/way, for a unified cache implementation:

`0x0` None supported

L1HvdSW, [11:8]

Indicates the supported L1 cache line maintenance operations by set/way, for a Harvard cache implementation:

`0x0` None supported

L1UniVA, [7:4]

Indicates the supported L1 cache line maintenance operations by MVA, for a unified cache implementation:

`0x0` None supported

L1HvdVA, [3:0]

Indicates the supported L1 cache line maintenance operations by MVA, for a Harvard cache implementation:

`0x0` None supported

Configurations

Must be interpreted with ID_MMFR0_EL1, ID_MMFR2_EL1, ID_MMFR3_EL1, and ID_MMFR4_EL1. See:

- *B2.95 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0, EL1* on page B2-292
- *B2.97 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2, EL1* on page B2-296
- *B2.98 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3, EL1* on page B2-298
- *B2.99 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4, EL1* on page B2-300

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.97 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2, EL1

The ID_MMFR2_EL1 provides information about the implemented memory model and memory management support in AArch32.

Bit field descriptions

ID_MMFR2_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

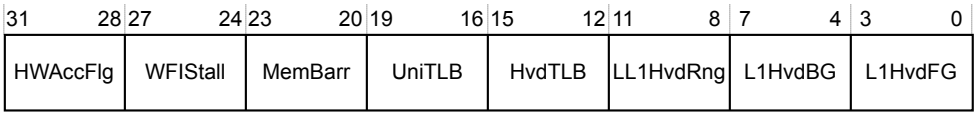

Figure B2-80 ID_MMFR2_EL1 bit assignments

HWAccFlg, [31:28]

Hardware access flag. Indicates support for a hardware access flag, as part of the VMSAv7 implementation:

`0x0` Not supported

WFIStall, [27:24]

Wait For Interrupt Stall. Indicates the support for *Wait For Interrupt* (WFI) stalling:

`0x1` Support for WFI stalling

MemBarr, [23:20]

Memory Barrier. Indicates the supported CP15 memory barrier operations.

`0x2` Supported CP15 memory barrier operations are:
- *Data Synchronization Barrier* (DSB)
- *Instruction Synchronization Barrier* (ISB)
- *Data Memory Barrier* (DMB)

UniTLB, [19:16]

Unified TLB. Indicates the supported TLB maintenance operations, for a unified TLB implementation.

`0x6` Supported unified TLB maintenance operations are:
- Invalidate all entries in the TLB
- Invalidate TLB entry by MVA
- Invalidate TLB entries by ASID match
- Invalidate instruction TLB and data TLB entries by MVA All ASID. This is a shared unified TLB operation.
- Invalidate Hyp mode unified TLB entry by MVA
- Invalidate entire Non-secure EL1 and EL0 unified TLB
- Invalidate entire Hyp mode unified TLB
- `TLBIMVALIS`, `TLBIMVAALIS`, `TLBIMVALHIS`, `TLBIMVAL`, `TLBIMVAAL`, and `TLBIMVALH`
- `TLBIIPAS2IS`, `TLBIIPAS2LIS`, `TLBIIPAS2`, and `TLBIIPAS2L`

HvdTLB, [15:12]

Harvard TLB. Indicates the supported TLB maintenance operations, for a Harvard TLB implementation:

`0x0` Not supported

LL1HvdRng, [11:8]

L1 Harvard cache Range. Indicates the supported L1 cache maintenance range operations, for a Harvard cache implementation:

`0x0` Not supported

L1HvdBG, [7:4]

L1 Harvard cache Background fetch. Indicates the supported L1 cache background prefetch operations, for a Harvard cache implementation:

`0x0` Not supported

L1HvdFG, [3:0]

L1 Harvard cache Foreground fetch. Indicates the supported L1 cache foreground prefetch operations, for a Harvard cache implementation:

`0x0` Not supported

Configurations

Must be interpreted with ID_MMFR0_EL1, ID_MMFR1_EL1, ID_MMFR3_EL1, and ID_MMFR4_EL1. See:

- *B2.95 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0, EL1* on page B2-292
- *B2.96 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1, EL1* on page B2-294
- *B2.98 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3, EL1* on page B2-298
- *B2.99 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4, EL1* on page B2-300

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.98 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3, EL1

The ID_MMFR3_EL1 provides information about the memory model and memory management support in AArch32.

Bit field descriptions

ID_MMFR3_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

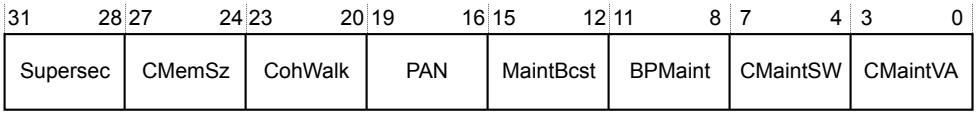

Figure B2-81 ID_MMFR3_EL1 bit assignments

Supersec, [31:28]

Supersections. Indicates support for supersections:

`0x0` Supersections supported

CMemSz, [27:24]

Cached memory size. Indicates the size of physical memory supported by the core caches:

`0x2` 1TByte or more, corresponding to a 40-bit or larger physical address range

CohWalk, [23:20]

Coherent walk. Indicates whether translation table updates require a clean to the point of unification:

`0x1` Updates to the translation tables do not require a clean to the point of unification to ensure visibility by subsequent translation table walks.

PAN, [19:16]

Privileged Access Never

`0x2`

PAN supported and new `ATS1CPRP` and `ATS1CPWP` instructions supported

MaintBcst, [15:12]

Maintenance broadcast. Indicates whether cache, TLB, and branch predictor operations are broadcast:

`0x2` Cache, TLB, and branch predictor operations affect structures according to shareability and defined behavior of instructions.

BPMaint, [11:8]

Branch predictor maintenance. Indicates the supported branch predictor maintenance operations:

`0x2` Supported branch predictor maintenance operations are:
- Invalidate all branch predictors
- Invalidate branch predictors by MVA

CMaintSW, [7:4]

Cache maintenance by set/way. Indicates the supported cache maintenance operations by set/way:

`0x1` Supported hierarchical cache maintenance operations by set/way are:
- Invalidate data cache by set/way
- Clean data cache by set/way
- Clean and invalidate data cache by set/way

CMaintVA, [3:0]

Cache maintenance by *Virtual Address* (VA). Indicates the supported cache maintenance operations by VA:

`0x1` Supported hierarchical cache maintenance operations by VA are:
- Invalidate data cache by VA
- Clean data cache by VA
- Clean and invalidate data cache by VA
- Invalidate instruction cache by VA
- Invalidate all instruction cache entries

Configurations

Must be interpreted with ID_MMFR0_EL1, ID_MMFR1_EL1, ID_MMFR2_EL1, and ID_MMFR4_EL1. See:
- *B2.95 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0, EL1* on page B2-292
- *B2.96 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1, EL1* on page B2-294
- *B2.97 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2, EL1* on page B2-296
- *B2.99 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4, EL1* on page B2-300

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.99 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4, EL1

The ID_MMFR4_EL1 provides information about the memory model and memory management support in AArch32.

Bit field descriptions

ID_MMFR4_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

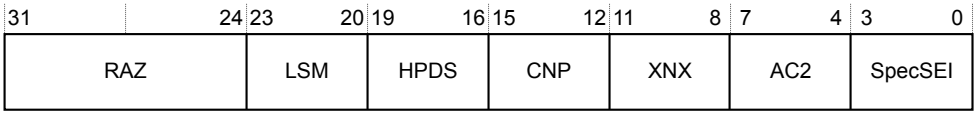

Figure B2-82 ID_MMFR4_EL1 bit assignments

RAZ, [31:24]

Read-As-Zero

LSM, [23:20]

Load/Store Multiple. Indicates whether adjacent loads or stores can be combined. The value is:

`0x0` LSMAOE and nTLSMD bit not supported

HPDS, [19:16]

Presence of Hierarchical Disables. Enables an operating system or hypervisor to hand over up to 4 bits of the last level page table descriptor (bits[62:59] of the page table entry) for use by hardware for IMPLEMENTATION DEFINED usage. The value is:

`0x2` Hierarchical Permission Disables and Hardware allocation of bits[62:59] supported

CNP, [15:12]

Common Not Private. Indicates support for selective sharing of TLB entries across multiple PEs. The value is:

`0x1` CnP bit supported

XNX, [11:8]

Execute-never. Indicates whether the stage 2 translation tables allows the stage 2 control of whether memory is executable at EL1 independent of whether memory is executable at EL0. The value is:

`0x1` EL0/EL1 execute control distinction at stage2 bit supported

AC2, [7:4]

Indicates the extension of the ACTLR and HACTLR registers using ACTLR2 and HACTLR2. The value is:

`0x1` ACTLR2 and HACTLR2 implemented

SpecSEI, [3:0]

Describes whether the core can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The value is:

`0x0` The core never generates an SError interrupt due to an External abort on a speculative read.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Must be interpreted with ID_MMFR0_EL1, ID_MMFR1_EL1, ID_MMFR2_EL1, and ID_MMFR3_EL1. See:

- *B2.95 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0, EL1* on page B2-292
- *B2.96 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1, EL1* on page B2-294
- *B2.97 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2, EL1* on page B2-296
- *B2.98 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3, EL1* on page B2-298

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.100 ID_PFR0_EL1, AArch32 Processor Feature Register 0, EL1

The ID_PFR0_EL1 provides top-level information about the instruction sets supported by the core in AArch32.

Bit field descriptions

ID_PFR0_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

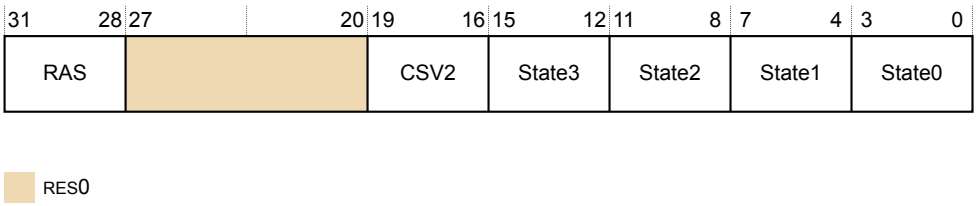

Figure B2-83 ID_PFR0_EL1 bit assignments

RAS, [31:28]

RAS extension version. The value is:

`0x1` Version 1 of the RAS extension is present.

RES0, [27:20]

RES0 Reserved

CSV2, [19:16]

`0x0` This device does not disclose whether branch targets trained in one context can affect speculative execution in a different context.

`0x1` Branch targets trained in one context cannot affect speculative execution in a different hardware described context. This is the reset value.

State3, [15:12]

Indicates support for *Thumb Execution Environment* (T32EE) instruction set. This value is:

`0x0` Core does not support the T32EE instruction set.

State2, [11:8]

Indicates support for Jazelle. This value is:

`0x1` Core supports trivial implementation of Jazelle.

State1, [7:4]

Indicates support for T32 instruction set. This value is:

`0x3` Core supports T32 encoding after the introduction of Thumb-2 technology, and for all 16-bit and 32-bit T32 basic instructions.

State0, [3:0]

Indicates support for A32 instruction set. This value is:

`0x1` A32 instruction set implemented

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.101 ID_PFR1_EL1, AArch32 Processor Feature Register 1, EL1

The ID_PFR1_EL1 provides information about the programmers model and architecture extensions supported by the core.

Bit field descriptions

ID_PFR1_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

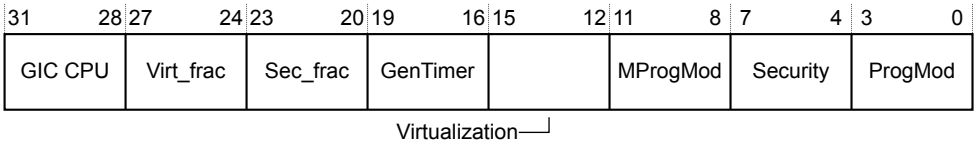

Figure B2-84 ID_PFR1_EL1 bit assignments

GIC CPU, [31:28]

GIC CPU support:

0 GIC CPU interface is disabled, **GICCDISABLE** is HIGH, or not implemented.

1 GIC CPU interface is implemented and enabled, **GICCDISABLE** is LOW.

Virt_frac, [27:24]

0 No features from the Armv7 Virtualization Extensions are implemented.

Sec_frac, [23:20]

0 No features from the Armv7 Virtualization Extensions are implemented.

GenTimer, [19:16]

Generic Timer support:

1 Generic Timer supported

Virtualization, [15:12]

Virtualization support:

0 Virtualization not implemented

MProgMod, [11:8]

M profile programmers model support:

0 Not supported

Security, [7:4]

Security support:

0 Security not implemented

ProgMod, [3:0]

Indicates support for the standard programmers model for Armv4 and later.

Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined, and System modes:

0 Not supported

Configurations

There are no configuration notes.

B2.102 ID_PFR2_EL1, AArch32 Processor Feature Register 2, EL1

The ID_PFR2_EL1 provides information about the programmers model and architecture extensions supported by the core.

Bit field descriptions

ID_PFR2_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

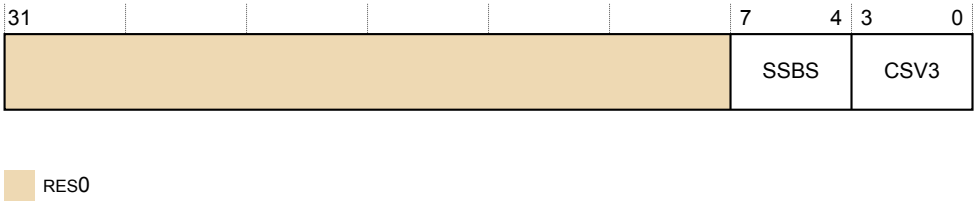

Figure B2-85 ID_PFR2_EL1 bit assignments

RES0, [31:8]

RES0 Reserved

SSBS, [7:4]

1 AArch32 provides the PSTATE.SSBS mechanism to mark regions that are *Speculative Store Bypassing Safe* (SSBS).

CSV3, [3:0]

1 Data loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes to be used by instructions newer than the load in the speculative sequence. This is the reset value.

Configurations

There are no configuration notes.

B2.103 LORC_EL1, LORegion Control Register, EL1

The LORC_EL1 register enables and disables LORegions, and selects the current LORegion descriptor.

Bit field descriptions

LORC_EL1 is a 64-bit register and is part of the Virtual memory control registers functional group.

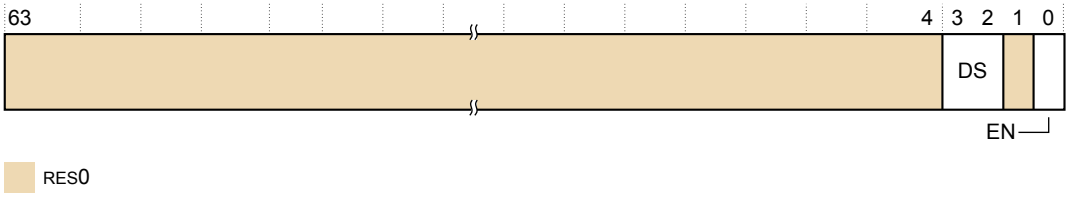

Figure B2-86 LORC_EL1 bit assignments

RES0, [63:4]

RES0 Reserved

DS, [3:2]

Descriptor Select. Number that selects the current LORegion descriptor accessed by the LORSA_EL1, LOREA_EL1, and LORN_EL1 registers.

RES0, [1]

RES0 Reserved

EN, [0]

Enable. The possible values are:

`0` Disabled. This is the reset value.

`1` Enabled

Configurations

RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.104 LORID_EL1, LORegion ID Register, EL1

The LORID_EL1 ID register indicates the supported number of LORegions and LORegion descriptors.

Bit field descriptions

LORID_EL1 is a 64-bit register.

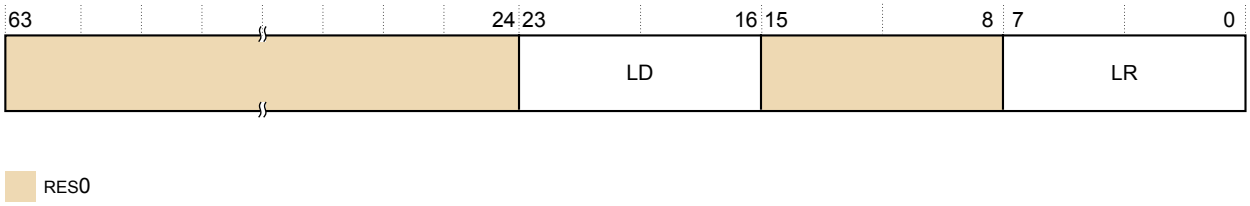

Figure B2-87 LORID_EL1 bit assignments

RES0, [63:24]

RES0 Reserved

LD, [23:16]

Number of LORegion descriptors supported by the implementation, expressed as binary 8-bit number. The value is:

`0x04` Four LORegion descriptors supported

RES0, [15:8]

RES0 Reserved

LR, [7:0]

Number of LORegions supported by the implementation, expressed as a binary 8-bit number. The value is:

`0x04` Four LORegions supported

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.105 LORN_EL1, LORegion Number Register, EL1

The LORN_EL1 register holds the number of the LORegion described in the current LORegion descriptor selected by LORC_EL1.DS.

Bit field descriptions

LORN_EL1 is a 64-bit register and is part of the Virtual memory control registers functional group.

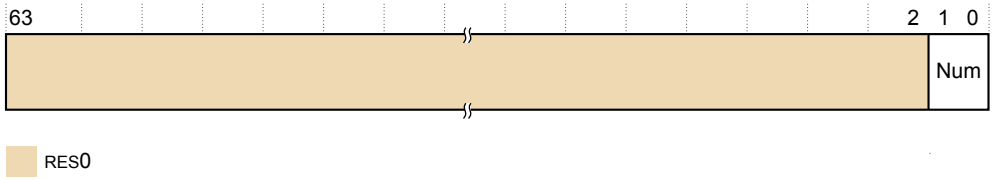

Figure B2-88 LORN_EL1 bit assignments

RES0, [63:2]

RES0 Reserved

Num, [1:0]

Indicates the LORegion number

Configurations

RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.106 MDCR_EL3, Monitor Debug Configuration Register, EL3

The MDCR_EL3 provides configuration options for Security to self-hosted debug.

Bit field descriptions

MDCR_EL3 is a 32-bit register, and is part of:

- The Debug registers functional group
- The Security registers functional group

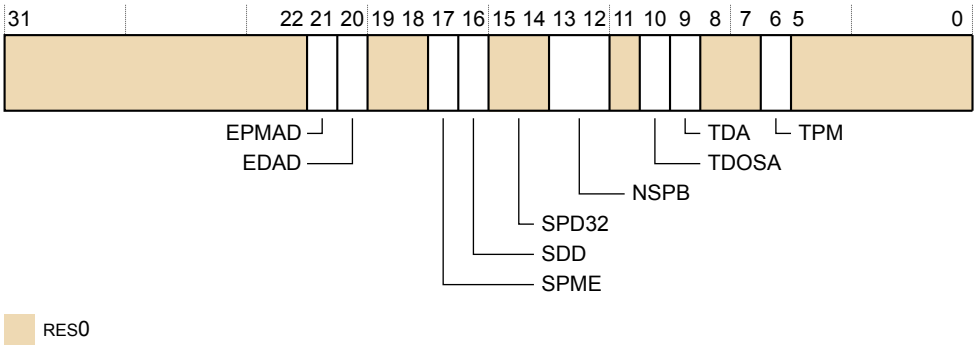

Figure B2-89 MDCR_EL3 bit assignments

RES0, [31:22]

RES0 Reserved

EPMAD, [21]

External debugger access to Performance Monitors registers disabled. This disables access to these registers by an external debugger. The possible values are:

0 Access to Performance Monitors registers from external debugger is permitted.

1 Access to Performance Monitors registers from external debugger is disabled, unless overridden by authentication interface.

EDAD, [20]

External debugger access to breakpoint and watchpoint registers disabled. This disables access to these registers by an external debugger. The possible values are:

0 Access to breakpoint and watchpoint registers from external debugger is permitted.

1 Access to breakpoint and watchpoint registers from external debugger is disabled, unless overridden by authentication interface.

SPME, [17]

Secure performance monitors enable. This enables event counting exceptions from Secure state. The possible values are:

0 Event counting prohibited in Secure state

1 Event counting allowed in Secure state

SPD32, [15:14]

RES0

Reserved

NSPB, [13:12]

Non-secure Profiling Buffer. Controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control registers. The possible values are:

`00` Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer controls at EL2 and EL1 in both security states generate Trap exceptions to EL3.

`01` Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer controls in Non-secure state generate Trap exceptions to EL3.

`10` Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer controls at EL2 and EL1 in both security states generate Trap exceptions to EL3.

`11` Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer controls at Secure EL1 generate Trap exceptions to EL3.

RES0, [11]

RES0 Reserved

TDOSA, [10]

Trap accesses to the OS debug system registers, OSLAR_EL1, OSLSR_EL1, OSDLR_EL1, and DBGPRCR_EL1 OS.

`0` Accesses are not trapped.

`1` Accesses to the OS debug system registers are trapped to EL3.

The reset value is UNKNOWN.

TDA, [9]

Trap accesses to the remaining sets of debug registers to EL3.

`0` Accesses are not trapped.

`1` Accesses to the remaining debug system registers are trapped to EL3.

The reset value is UNKNOWN.

RES0, [8:7]

RES0 Reserved

TPM, [6]

Trap Performance Monitors accesses. The possible values are:

`0` Accesses are not trapped.

`1` Accesses to the Performance Monitor registers are trapped to EL3.

The reset value is UNKNOWN.

RES0, [5:0]

RES0 Reserved

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.107 MIDR_EL1, Main ID Register, EL1

The MIDR_EL1 provides identification information for the core, including an implementer code for the device and a device ID number.

Bit field descriptions

MIDR_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

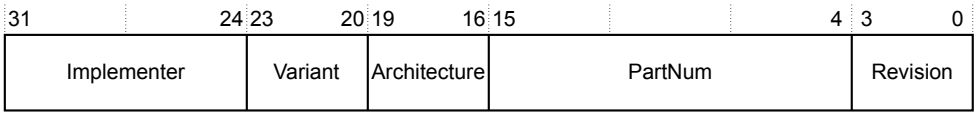

Figure B2-90 MIDR_EL1 bit assignments

Implementer, [31:24]

Indicates the implementer code. This value is:

`0x41` ASCII character 'A' - implementer is Arm Limited.

Variant, [23:20]

Indicates the variant number of the core. This is the major revision number *x* in the r*x* part of the r*x*p*y* description of the product revision status. This value is:

`0x0` r0p1

Architecture, [19:16]

Indicates the architecture code. This value is:

`0xF` Defined by CPUID scheme

PartNum, [15:4]

Indicates the primary part number. This value is:

`0xD4B` Cortex-A78C core

Revision, [3:0]

Indicates the minor revision number of the core. This is the minor revision number *y* in the p*y* part of the r*x*p*y* description of the product revision status. This value is:

`0x1` r0p1

Configurations

The MIDR_EL1 is architecturally mapped to external MIDR_EL1 register.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1

The MPIDR_EL1 provides an additional core identification mechanism for scheduling purposes in a cluster.

Bit field descriptions

MPIDR_EL1 is a 64-bit register, and is part of the Other system control registers functional group.

This register is read-only.

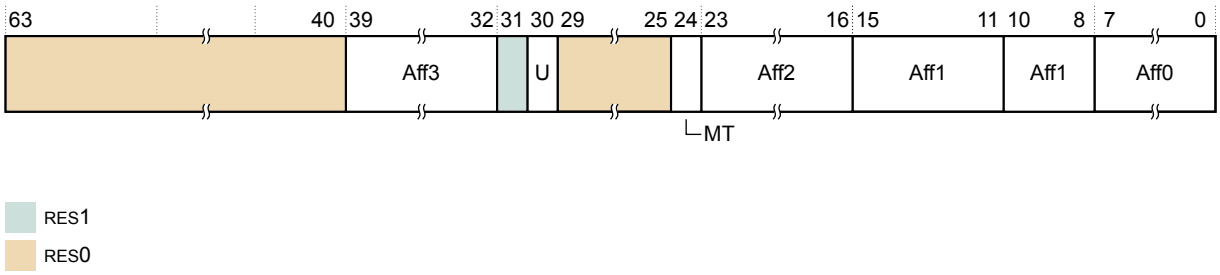

Figure B2-91 MPIDR_EL1 bit assignments

RES0, [63:40]

RES0 Reserved

Aff3, [39:32]

Affinity level 3. Highest level affinity field.

CLUSTERID
Indicates the value read in the **CLUSTERIDAFF3** configuration signal.

RES1, [31]

RES1 Reserved

U, [30]

Indicates a single core system, as distinct from core 0 in a cluster. This value is:

`0` Core is part of a multiprocessor system. This is the value for implementations with more than one core, and for implementations with an ACE or CHI master interface.

RES0, [29:25]

RES0 Reserved

MT, [24]

Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multithreading type approach. This value is:

`1` Performance of PEs at the lowest affinity level is very interdependent.

Affinity0 represents threads. Cortex-A78C is not multithreaded, but may be in a system with other cores that are multithreaded.

Aff2, [23:16]

Affinity level 2. Second highest level affinity field.

CLUSTERID
Indicates the value read in the **CLUSTERIDAFF2** configuration signal.

Aff1, [15:11]
Part of Affinity level 1. Third highest level affinity field.

RAZ Read-As-Zero

Aff1, [10:8]

Part of Affinity level 1. Third highest level affinity field.

CPUID Identification number for each CPU in the cluster:

`0x0` MP1: CPUID: 0

to

`0x7` MP8: CPUID: 7

Aff0, [7:0]
Affinity level 0. The level identifies individual threads within a multithreaded core. The Cortex-A78C core is single-threaded, so this field has the value `0x00`.

Configurations

MPIDR_EL1[31:0] is mapped to external register EDDEVAFF0.

MPIDR_EL1[63:32] is mapped to external register EDDEVAFF1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.109 PAR_EL1, Physical Address Register, EL1

The PAR_EL1 returns the output address from an address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

Bit field descriptions, PAR_EL1.F is 0

The following figure shows the PAR bit assignments when PAR.F is 0.

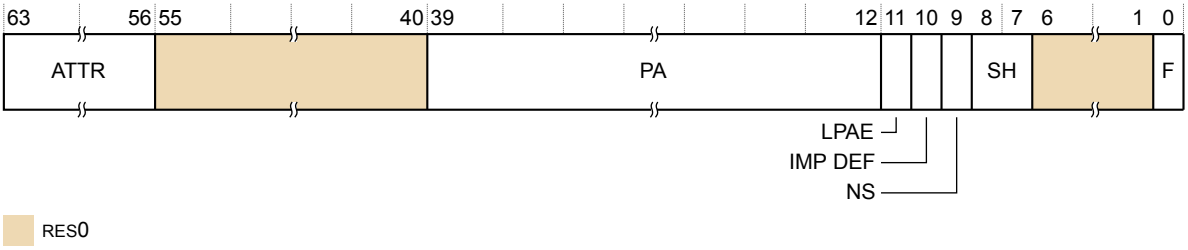

Figure B2-92 PAR bit assignments, PAR_EL1.F is 0

IMP DEF, [10]

IMPLEMENTATION DEFINED. Bit[10] is RES0.

F, [0]

Indicates whether the instruction performed a successful address translation.

`0` Address translation completed successfully

`1` Address translation aborted

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Bit field descriptions, PAR_EL1.F is 1

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

B2.110 REVIDR_EL1, Revision ID Register, EL1

The REVIDR_EL1 provides revision information, additional to MIDR_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the Cortex-A78C core.

Bit field descriptions

REVIDR_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is read-only.

Figure B2-93 REVIDR_EL1 bit assignments

IMPLEMENTATION DEFINED, [31:0]

IMPLEMENTATION DEFINED

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.111 RMR_EL3, Reset Management Register

The RMR_EL3 controls the Execution state that the core boots into and allows request of a Warm reset.

Bit field descriptions

RMR_EL3 is a 32-bit register, and is part of the Reset management registers functional group.

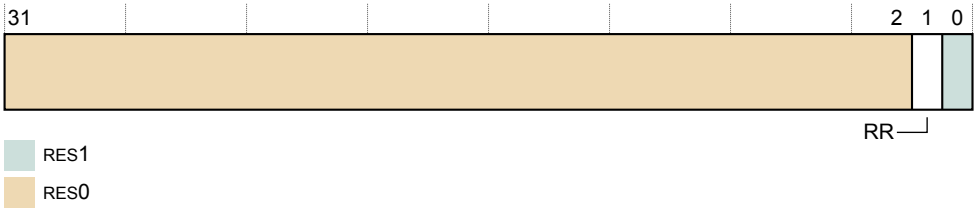

Figure B2-94 RMR_EL3 bit assignments

RES0, [31:2]

RES0 Reserved

RR, [1]

Reset Request. The possible values are:

0 This is the reset value on both a Warm and a Cold reset.

1 Requests a Warm reset.

The bit is strictly a request.

RES1, [0]

RES1 Reserved

Configurations

There are no configuration notes.

Details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.112 RVBAR_EL3, Reset Vector Base Address Register, EL3

RVBAR_EL3 contains the IMPLEMENTATION DEFINED address that execution starts from after reset.

Bit field descriptions

RVBAR_EL3 is a 64-bit register, and is part of the Reset management registers functional group.

This register is read-only.

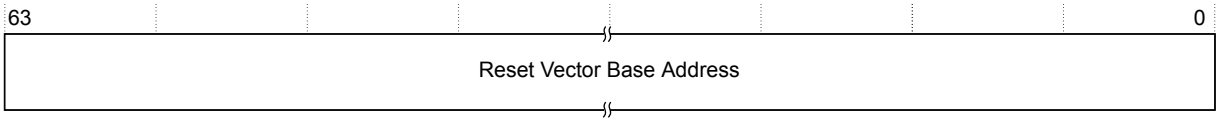

Figure B2-95 RVBAR_EL3 bit assignments

RVBA, [63:0]

Reset Vector Base Address. The address that execution starts from after reset when executing in 64-bit state. Bits[1:0] of this register are `0b00`, as this address must be aligned, and bits [63:40] are `0x000000` because the address must be within the physical address size supported by the core.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.113 SCTLR_EL1, System Control Register, EL1

The SCTLR_EL1 provides top-level control of the system, including its memory system, at EL1 and EL0.

Bit field descriptions

SCTLR_EL1 is a 64-bit register, and is part of the Other system control registers functional group.

This register resets to `0x0000000030D50838`.

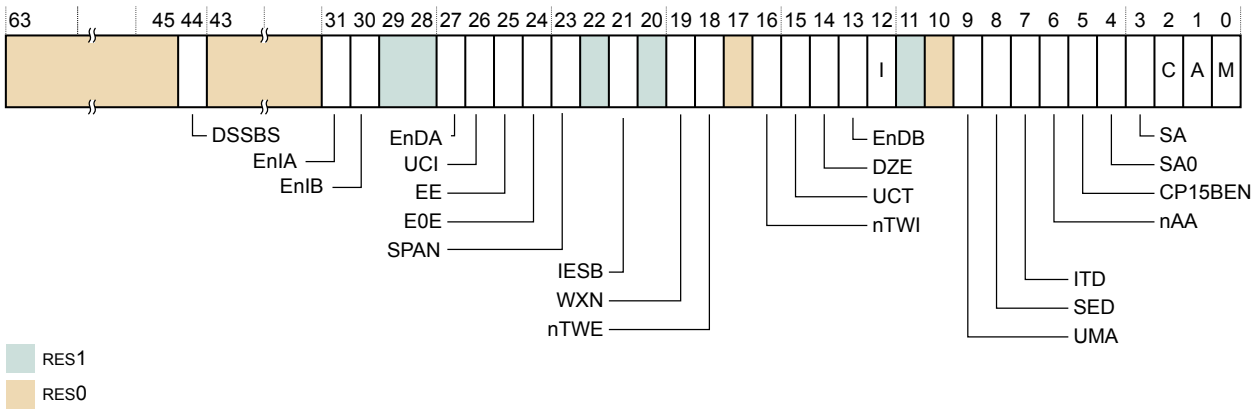

Figure B2-96 SCTLR_EL1 bit assignments

RES0, [63:45]

RES0 Reserved

DSSBS, [44]

DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).

`0x0` PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value.

`0x1` PSTATE.SSBS is set to 1 on an exception taken to this Exception level.

RES0, [43:32]

RES0 Reserved

EnIA, [31]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

EnIB, [30]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

RES1, [29:28]

RES1 Reserved

EnDA, [27]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

0 Pointer authentication of instruction addresses is not enabled.

1 Pointer authentication of instruction addresses is enabled.

EE, [25]

Exception endianness. The value of this bit controls the endianness for explicit data accesses at EL1. This value also indicates the endianness of the translation table data for translation table lookups. The possible values of this bit are:

0 Little-endian

1 Big-endian

IESB, [21]

Implicit error synchronization event enable. Possible values are:

0 Disabled

1 An implicit error synchronization event is added:
- After each exception taken to EL1
- Before the operational pseudocode of each ERET instruction executed at EL1

EnDB, [13]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

0 Pointer authentication of instruction addresses is not enabled.

1 Pointer authentication of instruction addresses is enabled.

ITD, [7]

This field is RAZ/WI.

nAA, [6]

When ARMv8.4-LSE is implemented, this bit controls generation of Alignment faults at EL1 and EL0 under certain conditions. The possible values are:

0b0 LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAPURW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for accesses.

0b1 This control bit does not cause LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAPURW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, or STLURH to generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes.

This field resets to an architecturally UNKNOWN value.

CP15BEN, [5]

CP15 barrier enable. The possible values are:

0 CP15 barrier operations disabled. Their encodings are UNDEFINED.

1 CP15 barrier operations enabled.

M, [0]

MMU enable. The possible values are:

`0` EL1 and EL0 stage 1 MMU disabled

`1` EL1 and EL0 stage 1 MMU enabled

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.114 SCTLR_EL2, System Control Register, EL2

The SCTLR_EL2 provides top-level control of the system, including its memory system at EL2.

Bit field descriptions

SCTLR_EL2 is a 64-bit register, and is part of:

- The Virtualization registers functional group
- The Other system control registers functional group

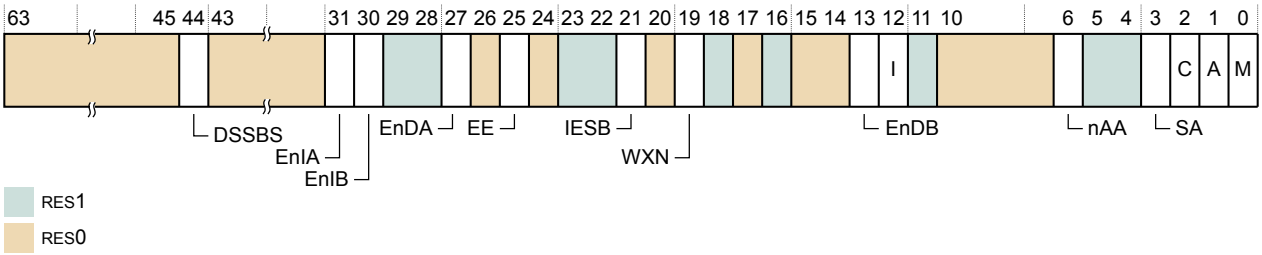

Figure B2-97 SCTLR_EL2 bit assignments

This register resets to `0x30C50838`.

DSSBS, [44]

DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).

SCTLR_EL2.DSSBS is held in bit[44] regardless of the value of HCR_EL2.E2H or HCR_EL2.TGE.

`0x0` PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value.

`0x1` PSTATE.SSBS is set to 1 on an exception taken to this Exception level.

EnIA, [31]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

EnIB, [30]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

EnDA, [27]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

IESB, [21]

When ARMv8.4-LSE is implemented, this bit controls generation of Alignment faults at EL1 and EL0 under certain conditions. The possible values are:

`0b0` Disabled

`0b1` An implicit error synchronization event is added:
- After each exception taken to EL2.
- Before the operational pseudocode of each ERET instruction executed at EL2.

EnDB, [13]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

nAA, [6]

When ARMv8.4-LSE is implemented, this bit controls generation of Alignment faults at EL1 and EL0 under certain conditions. The possible values are:

`0b0` LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAPURW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for accesses.

`0b1` This control bit does not cause LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAPURW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, or STLURH to generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes.

This field resets to an architecturally UNKNOWN value.

Configurations

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

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.115 SCTLR_EL3, System Control Register, EL3

The SCTLR_EL3 provides top-level control of the system, including its memory system at EL3.

Bit field descriptions

SCTLR_EL3 is a 64-bit register, and is part of the Other system control registers functional group.

This register resets to `0x30C50838`.

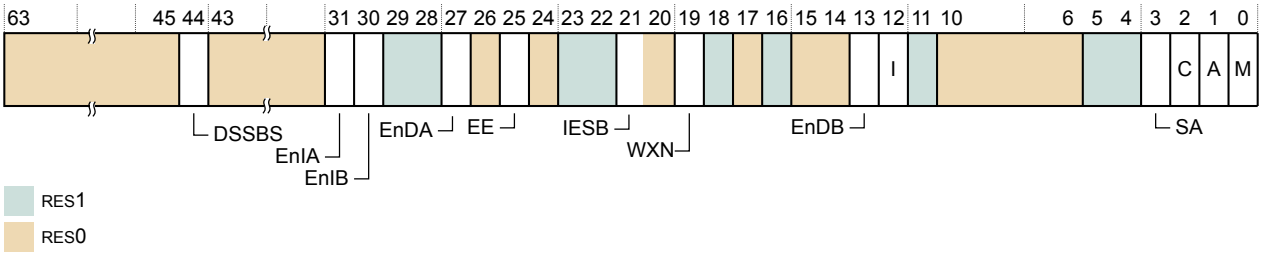

Figure B2-98 SCTLR_EL3 bit assignments

RES0, [63:45]

RES0 Reserved

DSSBS, [44]

DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).

`0x0` PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value.

`0x1` PSTATE.SSBS is set to 1 on an exception taken to this Exception level.

RES0, [43:32]

RES0 Reserved

EnIA, [31]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

EnIB, [30]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

RES1, [29:28]

RES1 Reserved

EnDA, [27]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

RES0, [26]

RES0 Reserved

EE, [25]

Exception endianness. This bit controls the endianness for:

- Explicit data accesses at EL3.
- Stage 1 translation table walks at EL3.

The possible values are:

`0x0` Little-endian

`0x1` Big-endian

The reset value is determined by the CFGEND configuration signal.

EnDB, [13]

Controls enabling of pointer authentication of instruction addresses in the EL1&0 translation regime. The possible values of this bit are:

`0` Pointer authentication of instruction addresses is not enabled.

`1` Pointer authentication of instruction addresses is enabled.

I, [12]

Global instruction cache enable. The possible values are:

`0x0` Instruction caches disabled. This is the reset value.

`0x1` Instruction caches enabled.

C, [2]

Global enable for data and unifies caches. The possible values are:

`0x0` Disables data and unified caches. This is the reset value.

`0x1` Enables data and unified caches.

M, [0]

Global enable for the EL3 MMU. The possible values are:

`0x0` Disables EL3 MMU. This is the reset value.

`0x1` Enables EL3 MMU.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.116 TCR_EL1, Translation Control Register, EL1

The TCR_EL1 determines which Translation Base registers define the base address register for a translation table walk required for stage 1 translation of a memory access from EL0 or EL1 and holds cacheability and shareability information.

Bit field descriptions

TCR_EL1 is a 64-bit register, and is part of the Virtual memory control registers functional group.

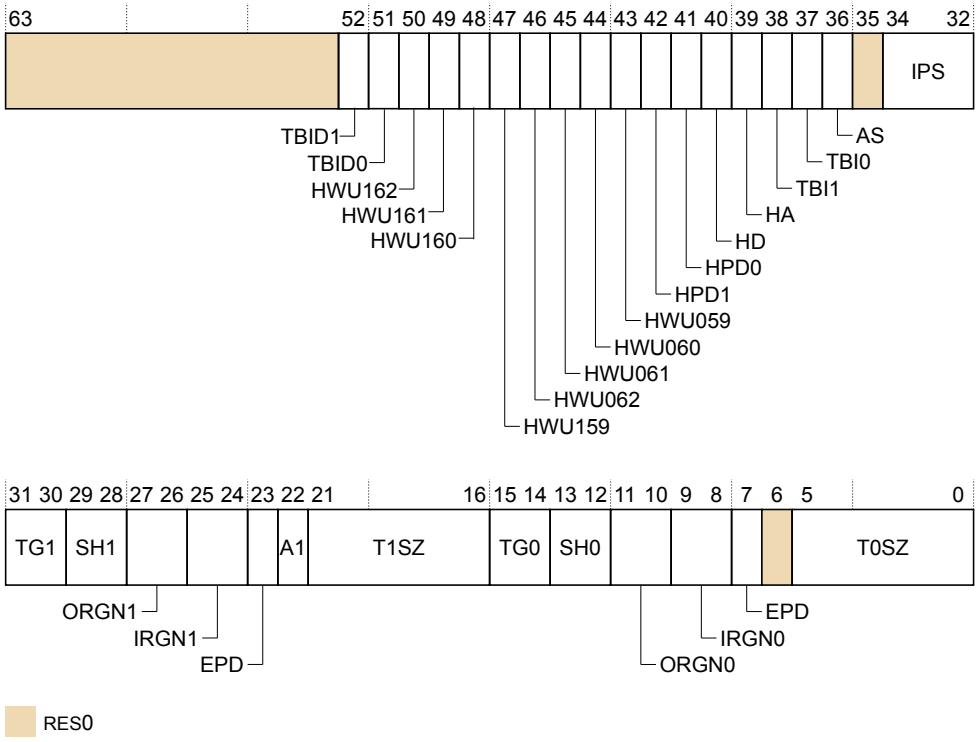

Figure B2-99 TCR_EL1 bit assignments

Note

Bits[50:39], architecturally defined, are implemented in the core.

TBID0, [52]

The possible values are:

`0` TBID0 applies to Instruction and Data accesses.

`1` TBID0 only applies to Data accesses.

TBID1, [51]

The possible values are:

`0` TBID1 applies to Instruction and Data accesses.

`1` TBID1 only applies to Data accesses.

HD, [40]

Hardware management of dirty state in stage 1 translations from EL0 and EL1. The possible values are:

`0` Stage 1 hardware management of dirty state disabled.

`1` Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

HA, [39]

Hardware Access flag update in stage 1 translations from EL0 and EL1. The possible values are:

`0` Stage 1 Access flag update disabled.

`1` Stage 1 Access flag update enabled.

Configurations

RW fields in this register reset to UNKNOWN values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.117 TCR_EL2, Translation Control Register, EL2 Primes

The TCR_EL2 controls translation table walks required for stage 1 translation of a memory access from EL2 and holds cacheability and shareability information.

Bit field descriptions

TCR_EL2 is a 64-bit register.

TCR_EL2 is part of:

- The Virtual memory control registers functional group.
- The Hypervisor and virtualization registers functional group.

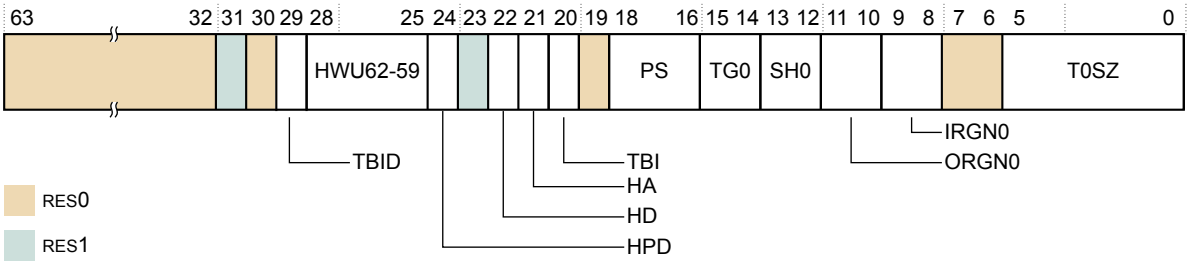

Figure B2-100 TCR_EL2 bit assignments

Note

Bits[28:21], architecturally defined, are implemented in the core.

TBID, [29]

Controls the use of the top byte of instruction addresses for address matching. The possible values are:

`0` TBI applies to Instruction and Data accesses.

`1` TBI applies to Data accesses only.

This field resets to an architecturally UNKNOWN value.

HD, [22]

Dirty bit update. The possible values are:

`0` Dirty bit update is disabled.

`1` Dirty bit update is enabled.

HA, [21]

Stage 1 Access flag update. The possible values are:

`0` Stage 1 Access flag update is disabled.

`1` Stage 1 Access flag update is enabled.

Configurations

When the Virtualization Host Extension is activated, TCR_EL2 has the same bit assignments as TCR_EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.118 TCR_EL3, Translation Control Register, EL3

The TCR_EL3 controls translation table walks required for stage 1 translation of memory accesses from EL3 and holds cacheability and shareability information for the accesses.

Bit field descriptions

TCR_EL3 is a 32-bit register and is part of the Virtual memory control registers functional group.

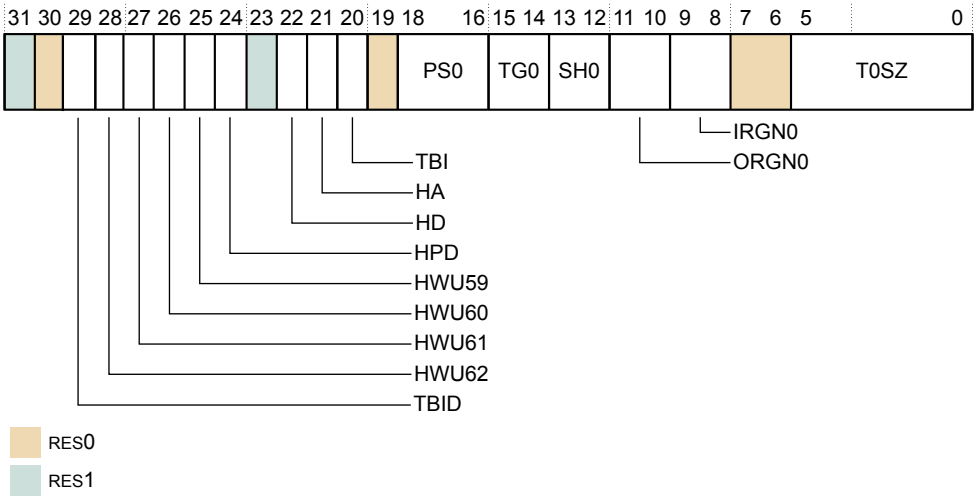

Figure B2-101 TCR_EL3 bit assignments

Note

Bits[28:21], architecturally defined, are implemented in the core.

TBID, [29]

Controls the use of the top byte of instruction addresses for address matching. The possible values are:

`0` TBI applies to Instruction and Data accesses.

`1` TBI applies to Data accesses only.

This field resets to an architecturally UNKNOWN value.

HD, [22]

Dirty bit update. The possible values are:

`0` Dirty bit update is disabled.

`1` Dirty bit update is enabled.

HA, [21]

Stage 1 Access flag update. The possible values are:

`0` Stage 1 Access flag update is disabled.

`1` Stage 1 Access flag update is enabled.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.119 TTBR0_EL1, Translation Table Base Register 0, EL1

The TTBR0_EL1 holds the base address of translation table 0, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses from modes other than Hyp mode.

Bit field descriptions

TTBR0_EL1 is 64-bit register.

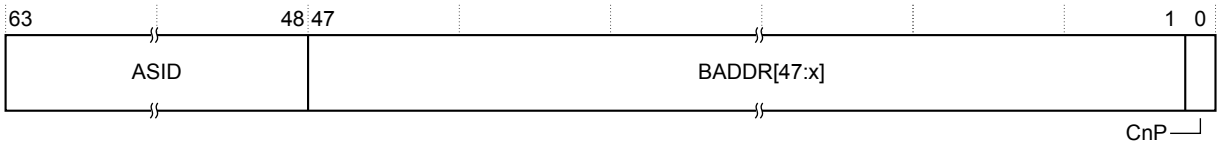

Figure B2-102 TTBR0_EL1 bit assignments

ASID, [63:48]

An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.

BADDR[47:x], [47:1]

Translation table base address, bits[47:x]. Bits [x-1:1] are RES0.

x is based on the value of TCR_EL1.T0SZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The value of x determines the required alignment of the translation table, that must be aligned to 2^x bytes.

If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.

CnP, [0]

Common not Private. The possible values are:

`0` CnP not supported

`1` CnP supported

Configurations

There are no configuration notes.
Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.120 TTBR0_EL2, Translation Table Base Register 0, EL2

The TTBR0_EL2 holds the base address of the translation table for the stage 1 translation of memory accesses from EL2.

Bit field descriptions

TTBR0_EL2 is a 64-bit register, and is part of the Virtual memory control registers functional group.

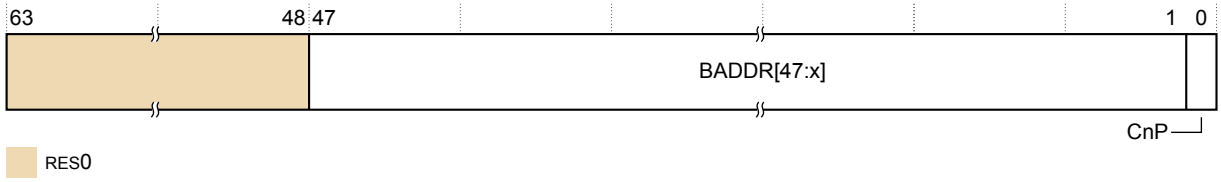

Figure B2-103 TTBR0_EL2 bit assignments

RES0, [63:48]

RES0 Reserved

BADDR, [47:1]

Translation table base address, bits[47:x]. Bits [x-1:1] are RES0.

x is based on the value of TCR_EL2.T0SZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Arm®v8, for Arm®v8-A architecture profile*.

The value of x determines the required alignment of the translation table, that must be aligned to 2^x bytes.

If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.

CnP, [0]

Common not Private. The possible values are:

`0` CnP not supported

`1` CnP supported

Configurations

When the Virtualization Host Extension is activated, TTBR0_EL2 has the same bit assignments as TTBR0_EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.121 TTBR0_EL3, Translation Table Base Register 0, EL3

The TTBR0_EL3 holds the base address of the translation table for the stage 1 translation of memory accesses from EL3.

Bit field descriptions

TTBR0_EL3 is a 64-bit register.

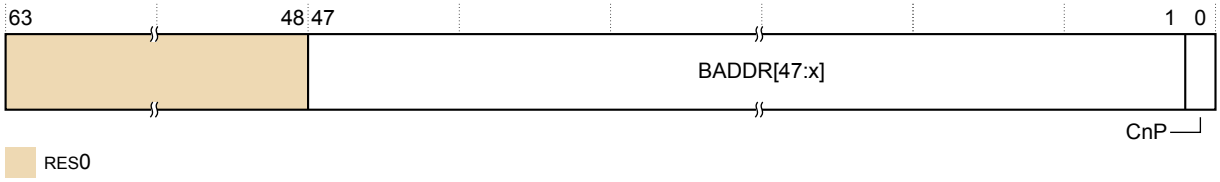

Figure B2-104 TTBR0_EL3 bit assignments

RES0, [63:48]

RES0 Reserved

BADDR[47:x], [47:1]

Translation table base address, bits[47:x]. Bits [x-1:1] are RES0.

x is based on the value of TCR_EL1.T0SZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The value of x determines the required alignment of the translation table, that must be aligned to 2^x bytes.

If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.

CnP, [0]

Common not Private. The possible values are:

`0` CnP not supported

`1` CnP supported

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.122 TTBR1_EL1, Translation Table Base Register 1, EL1

The TTBR1_EL1 holds the base address of translation table 1, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses at EL0 and EL1.

Bit field descriptions

TTBR1_EL1 is a 64-bit register.

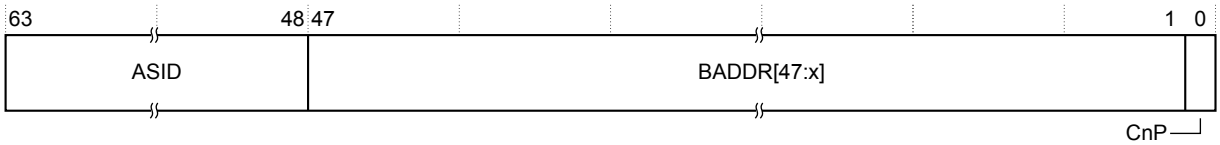

Figure B2-105 TTBR1_EL1 bit assignments

ASID, [63:48]

An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.

BADDR[47:x], [47:1]

Translation table base address, bits[47:x]. Bits [x-1:0] are RES0.

x is based on the value of TCR_EL1.T0SZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The value of x determines the required alignment of the translation table, that must be aligned to 2^x bytes.

If bits [x-1:1] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.

CnP, [0]

Common not Private. The possible values are:

`0` CnP not supported

`1` CnP supported

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.123 TTBR1_EL2, Translation Table Base Register 1, EL2

TTBR1_EL2 has the same format and contents as TTBR1_EL1.

See *B2.122 TTBR1_EL1, Translation Table Base Register 1, EL1* on page B2-335.

B2.124 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2

The VDISR_EL2 records that a virtual SError interrupt has been consumed by an `ESB` instruction executed at Non-secure EL1.

Bit field descriptions

VDISR_EL2 is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

Configurations

See *B2.125 VDISR_EL2 at EL1* on page B2-338.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.125 VDISR_EL2 at EL1

VDISR_EL2 has a specific format when written at EL1.

The following figure shows the VDISR_EL2 bit assignments when written at EL1:

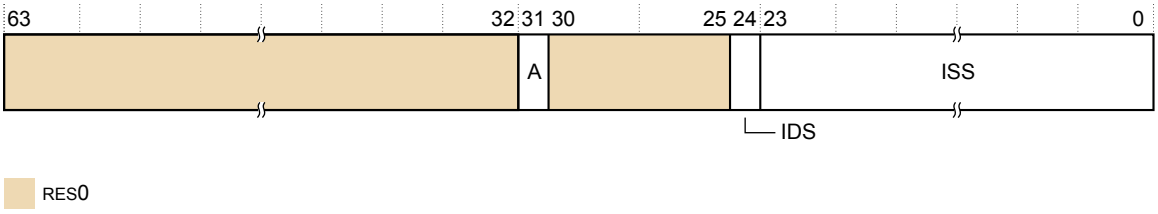

Figure B2-106 VDISR_EL2 at EL1 using AArch64

RES0, [63:32]

RES0 Reserved

A, [31]

Set to 1 when ESB defers an asynchronous SError interrupt.

RES0, [30:25]

RES0 Reserved

IDS, [24]

Contains the value from VSESR_EL2.IDS

ISS, [23:0]

Contains the value from VSESR_EL2, bits[23:0]

B2.126 VSESR_EL2, Virtual SError Exception Syndrome Register

The VSESR_EL2 provides the syndrome value reported to software on taking a virtual SError interrupt exception.

Bit field descriptions

VSESR_EL2 is a 64-bit register, and is part of:

- The Exception and fault handling registers functional group
- The Virtualization registers functional group

If the virtual SError interrupt is taken to EL1, VSESR_EL2 provides the syndrome value reported in ESR_EL1.

VSESR_EL2 bit assignments

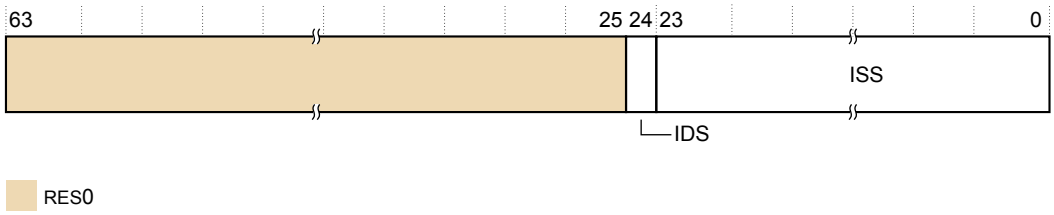

Figure B2-107 VSESR_EL2 bit assignments

RES0, [63:25]

RES0 Reserved

IDS, [24]

Indicates whether the deferred SError interrupt was of an IMPLEMENTATION DEFINED type. See ESR_EL1.IDS for a description of the functionality.

On taking a virtual SError interrupt to EL1 using AArch64 because HCR_EL2.VSE == 1, ESR_EL1[24] is set to VSESR_EL2.IDS.

ISS, [23:0]

Syndrome information. See ESR_EL1.ISS for a description of the functionality.

On taking a virtual SError interrupt to EL1 using AArch32 due to HCR_EL2.VSE == 1, ESR_EL1 [23:0] is set to VSESR_EL2.ISS.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.127 VTCR_EL2, Virtualization Translation Control Register, EL2

The VTCR_EL2 controls the translation table walks required for the stage 2 translation of memory accesses from Non-secure EL0 and EL1.

It also holds cacheability and shareability information for the accesses.

Bit field descriptions

VTCR_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group
- The Virtual memory control registers functional group

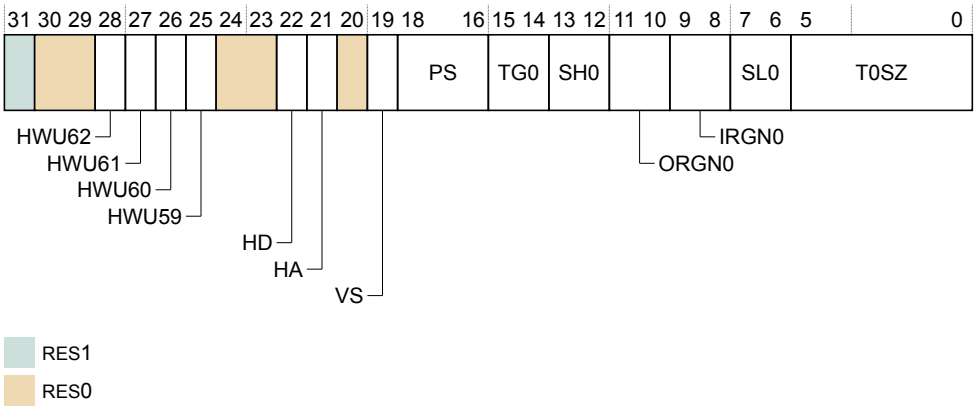

Figure B2-108 VTCR_EL2 bit assignments

Note

Bits[28:25] and bits[22:21], architecturally defined, are implemented in the core.

TG0, [15:14]

TTBR0_EL2 granule size. The possible values are:

`00` 4KB

`01` 64KB

`10` 16KB

`11` Reserved

All other values are not supported.

Configurations

RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

B2.128 VTTBR_EL2, Virtualization Translation Table Base Register, EL2

VTTBR_EL2 holds the base address of the translation table for the stage 2 translation of memory accesses from Non-secure EL0 and EL1.

Bit field descriptions

VTTBR_EL2 is a 64-bit register.

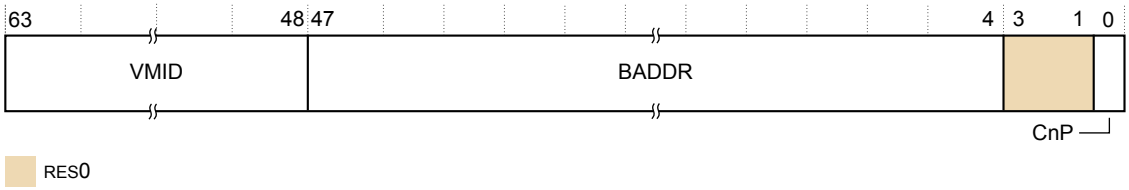

Figure B2-109 VTTBR_EL2 bit assignments

CnP, [0]

Common not Private. The possible values are:

`0` CnP not supported

`1` CnP supported

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Chapter B3
Error system registers

This chapter describes the error registers accessed by the AArch64 error registers.

It contains the following sections:

B3.1 Error system register summary

This section identifies the ERR0* core error record registers accessed by the AArch64 ERX* error registers.

For those registers that are not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The following table describes the architectural error record registers.

Table B3-1 Architectural error system register summary

Register mnemonic	Size	Register name	Access aliases from AArch64
ERR0ADDR	64	*B3.2 ERR0ADDR, Error Record Address Register* on page B3-345	*B2.61 ERXADDR_EL1, Selected Error Record Address Register, EL1* on page B2-242
ERR0CTLR	64	*B3.3 ERR0CTLR, Error Record Control Register* on page B3-346	*B2.62 ERXCTLR_EL1, Selected Error Record Control Register, EL1* on page B2-243
ERR0FR	64	*B3.4 ERR0FR, Error Record Feature Register* on page B3-348	*B2.63 ERXFR_EL1, Selected Error Record Feature Register, EL1* on page B2-244
ERR0MISC0	64	*B3.5 ERR0MISC0, Error Record Miscellaneous Register 0* on page B3-350	*B2.64 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1* on page B2-245
ERR0MISC1	64	*B3.6 ERR0MISC1, Error Record Miscellaneous Register 1* on page B3-355	*B2.65 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1, EL1* on page B2-246
ERR0STATUS	32	*B3.10 ERR0STATUS, Error Record Primary Status Register* on page B3-364	*B2.69 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1* on page B2-251

The following table describes the error record registers that are IMPLEMENTATION DEFINED.

Table B3-2 IMPLEMENTATION DEFINED error system register summary

Register mnemonic	Size	Register name	Access aliases from AArch64
ERR0PFGCDNR	32	*B3.7 ERR0PFGCDNR, Error Pseudo Fault Generation Count Down Register* on page B3-356	*B2.66 ERXPFGCDNR_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1* on page B2-247
ERR0PFGCTLR	32	*B3.8 ERR0PFGCTLR, Error Pseudo Fault Generation Control Register* on page B3-357	*B2.67 ERXPFGCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1* on page B2-248
ERR0PFGFR	32	*B3.9 ERR0PFGFR, Error Pseudo Fault Generation Feature Register* on page B3-361	*B2.68 ERXPFGFR_EL1, Selected Pseudo Fault Generation Feature Register, EL1* on page B2-250

B3.2 ERR0ADDR, Error Record Address Register

The ERR0ADDR stores the address that is associated to an error that is recorded.

Bit field descriptions

ERR0ADDR is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

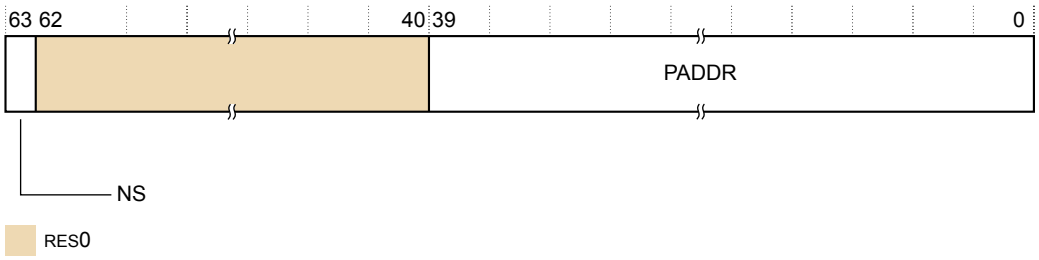

Figure B3-1 ERR0ADDR bit assignments

NS, [63]

Non-secure attribute. The possible values are:

`0` The physical address is Secure.

`1` The physical address is Non-secure.

RES0, [62:40]

RES0 Reserved

PADDR, [39:0]

Physical address

Configurations

ERR0ADDR resets to UNKNOWN.

When ERRSELR_EL1.SEL==0, this register is accessible from *B2.61 ERXADDR_EL1, Selected Error Record Address Register, EL1* on page B2-242

B3.3 ERR0CTLR, Error Record Control Register

The ERR0CTLR contains enable bits for the node that writes to this record:

- Enabling error detection and correction
- Enabling an error recovery interrupt
- Enabling a fault handling interrupt
- Enabling error recovery reporting as a read or write error response

Bit field descriptions

ERR0CTLR is a 64-bit register and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

ERR0CTLR resets to CFI [8], FI [3], and UI [2] are UNKNOWN. The rest of the register is RES0.

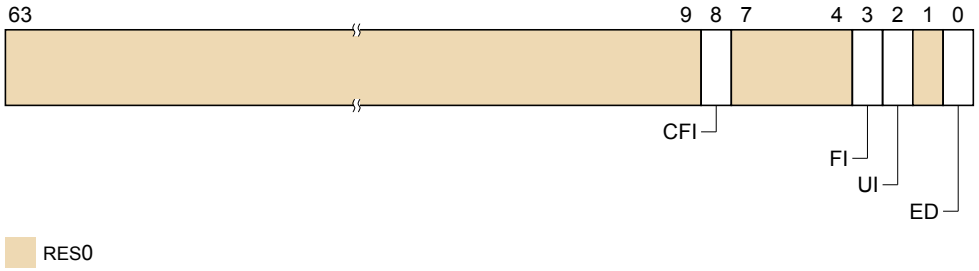

Figure B3-2 ERR0CTLR bit assignments

RES0, [63:9]

RES0 Reserved

CFI, [8]

Fault handling interrupt for corrected errors enable.

The fault handling interrupt is generated when one of the standard CE counters on ERR0MISC0 overflows and the overflow bit is set. The possible values are:

`0` Fault handling interrupt not generated for corrected errors.

`1` Fault handling interrupt generated for corrected errors.

The interrupt is generated even if the error status is overwritten because the error record already records a higher priority error.

Note

This applies to both reads and writes.

RES0, [7:4]

RES0 Reserved

FI, [3]

Fault handling interrupt enable.

The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are:

`0` Fault handling interrupt disabled

`1` Fault handling interrupt enabled

UI, [2]

Uncorrected error recovery interrupt enable. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. The possible values are:

`0` Error recovery interrupt disabled

`1` Error recovery interrupt enabled

Note

Applies to both reads and writes.

RES0, [1]

RES0 Reserved

ED, [0]

Error Detection and correction enable. The possible values are:

`0` Error detection and correction disabled

`1` Error detection and correction enabled

Configurations

This register is accessible from the following registers when ERRSELR_EL1.SEL==0:

B2.62 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page B2-243

B3.4 ERR0FR, Error Record Feature Register

The ERR0FR defines which of the common architecturally defined features are implemented and, of the implemented features, which are software programmable.

Bit field descriptions

ERR0FR is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

The register is read-only.

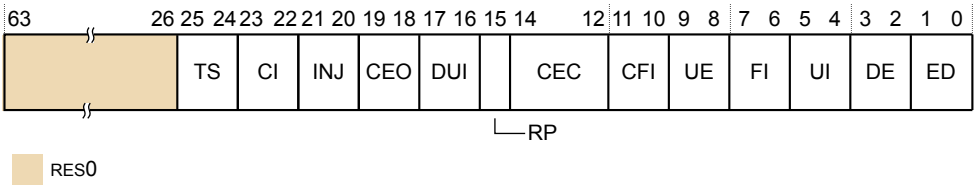

Figure B3-3 ERR0FR bit assignments

RES0, [63:26]

RES0 Reserved

TS, [25:24]

Timestamp Extension. The value is:

0b00 The node does not support a timestamp register.

CI, [23:22]

Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. The value is:

0b00 Does not support feature.

INJ, [21:20]

Fault Injection Extension. Indicates whether the standard fault injection mechanism is implemented. The value is:

0b00 The node does not support a standard fault injection mechanism.

CEO, [19:18]

Corrected Error Overwrite. The value is:

0b00 Counts CE if a counter is implemented and keeps the previous error status. If the counter overflows, ERR0STATUS.OF is set to 1.

DUI, [17:16]

Error recovery interrupt for deferred errors. The value is:

0b00 The core does not support this feature.

RP, [15]

Repeat counter. The value is:

`0b1` A first repeat counter and a second other counter are implemented. The repeat counter is the same size as the primary error counter.

CEC, [14:12]

Corrected Error Counter. The value is:

`0b010` The node implements an 8-bit standard CE counter in ERR0MISC0[39:32].

CFI, [11:10]

Fault handling interrupt for corrected errors. The value is:

`0b10` The node implements a control for enabling fault handling interrupts on corrected errors.

UE, [9:8]

In-band uncorrected error reporting. The value is:

`0b01` The node implements in-band uncorrected error reporting, that is External aborts.

FI, [7:6]

Fault handling interrupt. The value is:

`0b10` The node implements a fault handling interrupt and implements controls for enabling and disabling.

UI, [5:4]

Error recovery interrupt for uncorrected errors. The value is:

`0b10` The node implements an error recovery interrupt and implements controls for enabling and disabling.

DE, [3:2]

Defers errors. The value is:

`0b00`

Defers errors is always enabled for the node.

ED, [1:0]

Error detection and correction. The value is:

`0b10` The node implements controls for enabling or disabling error detection and correction.

Configurations

ERR0FR resets to `0x000000000000A9A2`

ERR0FR is accessible from the following registers when ERRSELR_EL1.SEL==0:

B2.63 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B2-244.

B3.5 ERR0MISC0, Error Record Miscellaneous Register 0

The ERR0MISC0 is an error syndrome register. It contains corrected error counters, information to identify where the error was detected, and other state information not present in the corresponding status and address error record registers.

Bit field descriptions

ERR0MISC0 is a 64-bit register, and is part of the *Reliability, Availability, Serviceability* (RAS) registers functional group.

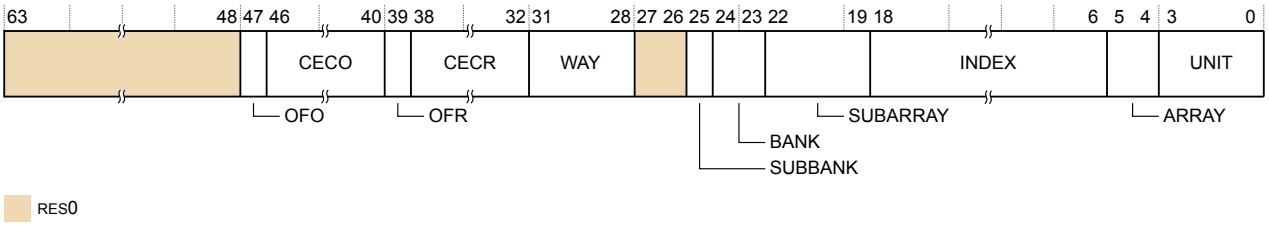

Figure B3-4 ERR0MISC0 bit assignments

RES0, [63:48]

RES0 Reserved

OFO, [47]

Sticky overflow bit, other. The possible values of this bit are:

0 Other counter has not overflowed.

1 Other counter has overflowed.

The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.

CECO, [46:40]

Corrected error count, other. Incremented for each Corrected error that does not match the recorded syndrome.

This field resets to an IMPLEMENTATION DEFINED which might be UNKNOWN on a Cold reset. If the reset value is UNKNOWN, then the value of this field remains UNKNOWN until software initializes it.

OFR, [39]

Sticky overflow bit, repeat. The possible values of this bit are:

0 Repeat counter has not overflowed.

1 Repeat counter has overflowed.

The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.

CECR, [38:32]

Corrected error count, repeat. Incremented for the first recorded error, which also records other syndromes, and then again for each Corrected error that matches the recorded syndrome.

This field resets to an IMPLEMENTATION DEFINED which might be UNKNOWN on a Cold reset. If the reset value is UNKNOWN, then the value of this field remains UNKNOWN until software initializes it.

WAY, [31:28]

The encoding depends on the unit from which the error being recorded was detected. The possible values are:

L1 Instruction Cache Tag Indicates which way of the Instruction Cache TAG RAM detected the error. Upper two bits are unused.

L1 Instruction Data Cache Indicates which way of the Instruction Cache Data RAM detected the error. Upper two bits are unused.

L0 Mop Cache Indicates which way of the Mop Cache RAM detected the error. Upper two bits are unused.

L1 Data Cache Data Indicates which way of the L1 Data Cache Data RAM detected the error. Upper two bits are unused.

L1 Data Cache Tag Indicates which way of the Instruction Cache Tag RAM detected the error. Upper two bits are unused.

L2 TLB Indicates which way of the L2 TLB RAM detected the error. Upper 1 bit is unused.

L2 Tag Indicates the way of the L2 Tag RAM that detected the error. Upper 1 bit is unused.

L2 Data Unused

L2 TQ Unused

L2 CHI Slave Unused

RES0, [27:26]

RES0 Reserved

SUBBANK, [25]

The encoding depends on the unit from which the error being recorded was detected. The possible values are:

L1 Instruction Cache Indicates which subbank has the error, valid for Instruction Data Cache, unused otherwise.

BANK, [24:23]

The encoding depends on the unit from which the error being recorded was detected. The possible values are:

L1 Instruction Cache Tag Unused

L1 Instruction Data Cache Indicates which bank (bank0 - bank3) detected the error.

L0 Mop Cache Indicates which bank (bank0 - bank3) detected the error.

L1 Data Cache Data Unused

L1 Data Cache Tag Unused

L2 TLB Unused

L2 Tag Indicates which bank detected the error.

L2 Data Indicates which bank detected the error.

L2 TQ Indicates which bank detected the error.

L2 CHI Slave Indicates which bank detected the error.

SUBARRAY, [22:19]

The encoding depends on the unit from which the error being recorded was detected. The possible values are:

L1 Instruction Cache Tag Unused

L1 Instruction Data Cache Unused

L0 Mop Cache Unused

L1 Data Cache Data Indicates the word that detected the error. Upper 1 bit is unused.

- `0b000` Word0
- `0b001` Word1
- `0b010` Word2
- `0b011` Word3
- `0b100` Word4
- `0b101` Word5
- `0b110` Word6
- `0b111` Word7

L1 Data Cache Tag Indicates the bank that detected the error. Upper 1 bit is unused.

- `0b0000` bank0
- `0b0001` bank1

L2 TLB Unused

L2 Tag Unused

L2 Data Indicates which double word detected the error. Upper 1 bit is unused.

- `0b000` DW0
- `0b001` DW1
- `0b010` DW2
- `0b011` DW3
- `0b100` DW4
- `0b101` DW5
- `0b110` DW6
- `0b111` DW7

L2 TQ Indicates which double word detected the error. Upper 1 bit is unused.

- `0b000` DW0
- `0b001` DW1
- `0b010` DW2
- `0b011` DW3
- `0b100` DW4
- `0b101` DW5
- `0b110` DW6
- `0b111` DW7

L2 CHI Slave

- `0b0000` Copyback error response.
- `0b0001` Undeferrable error when ECC is disabled.

INDEX, [18:6]

The encoding depends on the unit from which the error being recorded was detected. The possible values are:

L1 Instruction Cache Tag Indicates which index of the cache reported the error. Upper bits are unused.

L1 Instruction Data Cache Indicates which index of the cache reported the error. Upper bits are unused.

L0 Mop Cache Indicates which index of the cache reported the error. Upper bits are unused.

L1 Data Cache Data Indicates which index of the cache reported the error. Upper bits are unused.

L1 Data Cache Tag Indicates which index of the cache reported the error. Upper bits are unused.

L2 TLB Indicates which index of the cache reported the error. Upper bits are unused.

L2 Tag Indicates which index of the cache reported the error. Upper bits are unused.

L2 Data Indicates which index of the data RAM reported the error. Software reading this error record must decode this field as follows, where L2_CACHE_SIZE_LOG = CLOG2(L2_CACHE_SIZE):

1. INDEX[18 : (L2_CACHE_SIZE_LOG + 3 + 6)] : RES0
2. INDEX[L2_CACHE_SIZE_LOG +2 + 6 : 6] : Indicates the index of the cache line that reported the error.

L2 TQ Indicates which entry of the L2TQ reported the error. Upper 8 bits are unused.

L2 CHI Slave Unused

ARRAY, [5:4]

The encoding depends on the unit from which the error being recorded was detected. The possible values are:

L2 Cache Indicates which array has the error. The possible values are:

`0b00` L2 Tag RAM

`0b01` L2 Data RAM

`0b10` TQ Data RAM

`0b11` CHI Slave Error

L1 Data Cache Indicates which array detected the error. The possible values are:

`0b00` LS0 copy of Tag RAM

`0b01` LS1 copy of Tag RAM

`0b10` LS Data RAM

`0b11` LS2 copy of Tag RAM

L1 Instruction Cache Indicates which array that detected the error, Data Array has higher priority. The possible values are:

`0b00` Tag

`0b01` Data

`0b10` Mop cache

UNIT, [3:0]

Indicates the unit which detected the error. The possible values are:

`0b0001` L1 Instruction Cache (Tag, Data, or Mop Cache)

`0b0010` L2 TLB

`0b0100` L1 Data Cache (Tag or Data)

`0b1000` L2 Cache (Tag, Data, TQ, or CHI slave)

Configurations

ERR0MISC0 resets to [63:32] is `0x00000000`, [31:0] is UNKNOWN.

This register is accessible from the following register when ERRSELR_EL1.SEL==0:

- *B2.64 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1* on page B2-245.

B3.6 ERR0MISC1, Error Record Miscellaneous Register 1

This register is unused in the Cortex-A78C core and marked as RES0.

Configurations

When ERRSELR_EL1.SEL==0, ERR0MISC1 is accessible from *B2.65 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1, EL1* on page B2-246.

B3.7 ERR0PFGCDNR, Error Pseudo Fault Generation Count Down Register

ERR0PFGCDNR is the Cortex-A78C node register that generates one of the errors that are enabled in the corresponding ERR0PFGCTLR register.

Bit field descriptions

ERR0PFGCDNR is a 32-bit register and is RW.

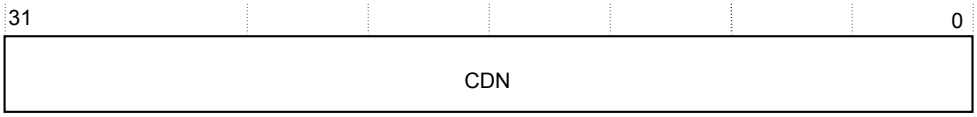

Figure B3-5 ERR0PFGCDNR bit assignments

CDN, [31:0]
Count Down value. The reset value of the Error Generation Counter is used for the countdown.

Configurations

There are no configuration options.

ERR0PFGCDNR resets to UNKNOWN.

When ERRSELR_EL1.SEL==0, ERR0PFGCDNR is accessible from *B2.66 ERXPFGCDNR_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1* on page B2-247.

B3.8 ERR0PFGCTLR, Error Pseudo Fault Generation Control Register

The ERR0PFGCTLR is the Cortex-A78C node register that enables controlled fault generation.

Bit field descriptions

ERR0PFGCTLR is a 32-bit read/write register.

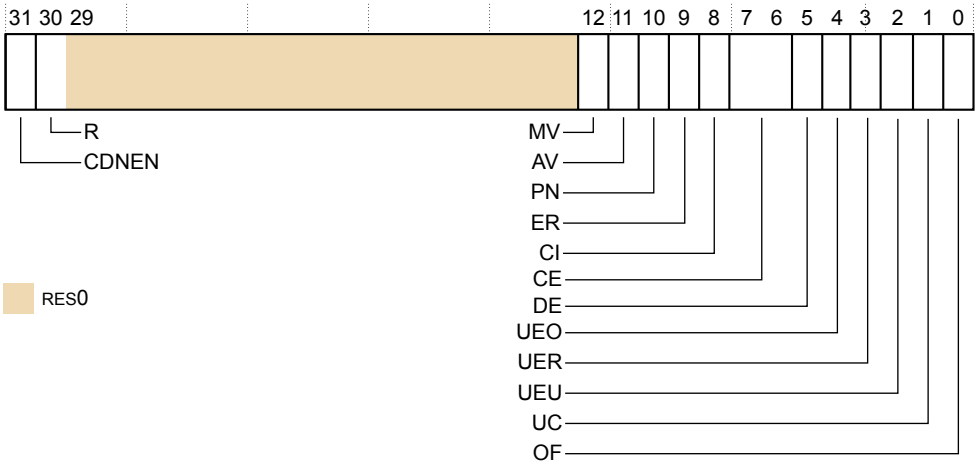

Figure B3-6 ERR0PFGCTLR bit assignments

CDNEN, [31]

Count down enable. This bit controls transfers from the value that is held in the ERR0PFGCDNR into the Error Generation Counter and enables this counter to start counting down. The possible values are:

`0` The Error Generation Counter is disabled.

`1` The value that is held in the ERR0PFGCDNR register is transferred into the Error Generation Counter. The Error Generation Counter counts down.

R, [30]

Restartable bit. When it reaches 0, the Error Generation Counter restarts from the ERR0PFGCDNR value or stops. The possible values are:

`0` When it reaches 0, the counter stops.

`1` When it reaches 0, the counter reloads the value that is stored in ERR0PFGCDNR and starts counting down again.

RES0, [29:13]

RES0 Reserved

MV, [12]

Miscellaneous syndrome. The value written to ERR<n>STATUS.MV when an injected error is recorded. The possible values of this bit are:

`0` ERR<n>STATUS.MV is set to 0 when an injected error is recorded.

`1` ERR<n>STATUS.MV is set to 0 when an injected error is recorded.

This bit reads-as-one if the node always records some syndrome in ERR<n>MISC<m>, setting ERR<n>STATUS.MV to 1, when an injected error is recorded. This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

AV, [11]

Address syndrome. The value written to ERR<n>STATUS.AV when an injected error is recorded. The possible values of this bit are:

`0` ERR<n>STATUS.AV is set to 0 when an injected error is recorded.

`1` ERR<n>STATUS.AV is set to 0 when an injected error is recorded.

This bit reads-as-one if the node always sets ERR<n>STATUS<m> to 1 when an injected error is recorded. This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

PN, [10]

Address syndrome. The value written to ERR<n>STATUS.PN when an injected error is recorded. The possible values of this bit are:

`0` ERR<n>STATUS.PN is set to 0 when an injected error is recorded.

`1` ERR<n>STATUS.PN is set to 0 when an injected error is recorded.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

ER, [9]

Address syndrome. The value written to ERR<n>STATUS.PN when an injected error is recorded. The possible values of this bit are:

`0` ERR<n>STATUS.ER is set to 0 when an injected error is recorded.

`1` ERR<n>STATUS.ER is set to 0 when an injected error is recorded.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

CI, [8]

Address syndrome. The value written to ERR<n>STATUS.CI when an injected error is recorded. The possible values of this bit are:

`0` ERR<n>STATUS.CI is set to 0 when an injected error is recorded.

`1` ERR<n>STATUS.CI is set to 0 when an injected error is recorded.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

CE, [7:6]

Corrected error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are:

`00` No corrected error is generated.

`01` A non-specific corrected error might be generated when the Error Generation Counter decrements to zero.

`10` A transient corrected error might be generated when the Error Generation Counter decrements to zero.

`11` A persistent corrected error might be generated when the Error Generation Counter decrements to zero.

The set of permitted values for this field is defined by ERR<n>PFGF.CE.

This field is RES0 if the node does not support this control.

This field resets to an architecturally UNKNOWN value on a Cold reset. This field is preserved on an Error Recovery reset.

DE, [5]

Deferred Error generation enable. The possible values are:

`0` No deferred error is generated.

`1` A deferred error might be generated when the Error Generation Counter is triggered.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

UEO, [4]

Latent or Restartable Error generation enable. Controls whether this type of error condition might be generated. It is IMPLEMENTATION DEFINED whether the error is generated if the data is not consumed. The possible values are:

`0` No error of this type will be generated.

`1` An error of this type might be generated when the Error Generation Counter decrements to zero.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

UER, [3]

Signaled or Recoverable Error generation enable. Controls whether this type of error condition might be generated. It is IMPLEMENTATION DEFINED whether the error is generated if the data is not consumed The possible values are:

`0` No error of this type will be generated.

`1` An error of this type might be generated when the Error Generation Counter decrements to zero.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

UEU, [2]

Unrecoverable Error generation enable. Controls whether this type of error condition might be generated. It is IMPLEMENTATION DEFINED whether the error is generated if the data is not consumed The possible values are:

`0` No error of this type will be generated.

`1` An error of this type might be generated when the Error Generation Counter decrements to zero.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

UC, [1]

Uncontainable error generation enable. The possible values are:

`0` No uncontainable error is generated.

`1` An uncontainable error might be generated when the Error Generation Counter is triggered.

OF, [0]

Overflow flag. The value written to ERR<n>STATUS.OF when an injected error is recorded. The possible values of this bit are:

`0` ERR<n>STATUS.OF is set to 0 when an injected error is recorded.

`1` ERR<n>STATUS.OF is set to 0 when an injected error is recorded.

This bit is RES0 if the node does not support this control.

This bit resets to an architecturally UNKNOWN value on a Cold reset. This bit is preserved on an Error Recovery reset.

Configurations

There are no configuration notes.

ERR0PFGCTLR resets to `0x00000000`.

ERR0PFGCTLR is accessible from the following registers when ERRSELR_EL1.SEL==0:

- *B2.67 ERXPFGCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1* on page B2-248

B3.9 ERR0PFGFR, Error Pseudo Fault Generation Feature Register

The ERR0PFGFR is the Cortex-A78C node register that defines which fault generation features are implemented.

Bit field descriptions

ERR0PFGFR is a 32-bit register and is RO.

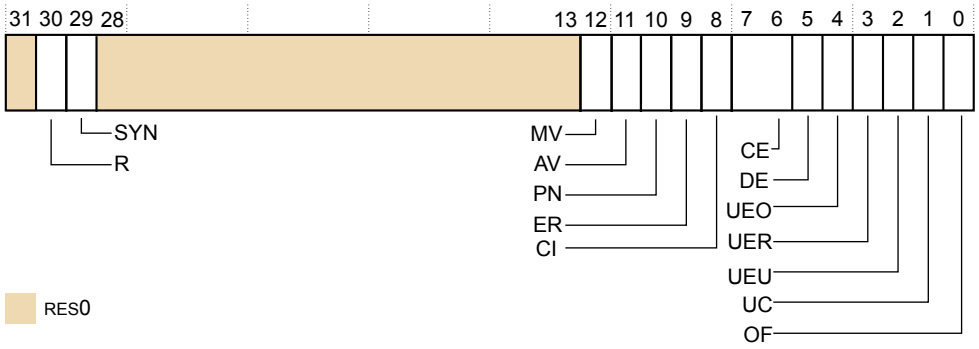

Figure B3-7 ERR0PFGFR bit assignments

RES0, [31]

RES0 Reserved

R, [30]

Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERR0PFGCDNR value or stops. The value is:

1 This feature is controllable.

SYN, [29]

Syndrome. Fault syndrome injection. The value is:

0 When an injected error is recorded, the node sets ERR<n>STATUS.{IERR, SERR} to IMPLEMENTATION DEFINED values. ERR<n>STATUS.{IERR, SERR} are UNKNOWN when ERR<n>STATUS.V is set to 0.

RES0, [28:13]

RES0 Reserved

MV, [12]

Miscellaneous syndrome. Additional syndrome injection. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. It is IMPLEMENTATION DEFINED which syndrome fields in ERR<n>MISC<m> this refers to, as some fields might always be recorded by an error, for example, a Corrected Error counter. The value is:

0 When an injected error is recorded, the node might record IMPLEMENTATION DEFINED additional syndrome in ERR<n>MISC<m>. If any syndrome is recorded in ERR<n>MISC<m>, then ERR<n>STATUS.MV is set to 1.

AV, [11]

Address syndrome. Address syndrome injection. The value is:

0 When an injected error is recorded, the node either sets ERR<n>ADDR and ERR<n>STATUS.AV for the access or leaves these unchanged.

PN, [10]

Poison flag. Describes how the fault generation feature of the node sets the ERR<n>STATUS.PN status flag. The value is:

0 When an injected error is recorded, the node sets ERR<n>STATUS.PN to 0.

ER, [9]

Error Reported flag. Describes how the fault generation feature of the node sets the ERR<n>STATUS.ER status flag. The value is:

0 When an injected error is recorded, the node sets ERR<n>STATUS.ER according to the architecture-defined rules for setting the ER bit.

CI, [8]

The value is:

0 When an injected error is recorded, it is IMPLEMENTATION DEFINED whether the node sets ERR<n>STATUS.CI to 1.

CE, [7:6]

Corrected Error generation. The value is:

1 This feature is controllable.

DE, [5]

Deferred Error generation. The value is:

1 This feature is controllable.

UEO, [4]

Latent or Restartable Error generation. The value is:

0 The node does not support this feature.

UER, [3]

Signaled or Recoverable Error generation. The value is:

0 The node does not support this feature.

UEU, [2]

Unrecoverable Error generation. The value is:

0 The node does not support this feature.

UC, [1]

Uncontainable Error generation. The value is:

1 This feature is controllable.

OF, [0]

Overflow flag. The value is:

0 When an injected error is recorded, the node sets ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF bit.

Configurations

There are no configuration notes.

ERR0PFGFR resets to `0xC0000062`.

When ERRSELR_EL1.SEL==0, ERR0PFGFR is accessible from *B2.68 ERXPFGFR_EL1, Selected Pseudo Fault Generation Feature Register, EL1* on page B2-250.

B3.10 ERR0STATUS, Error Record Primary Status Register

The ERR0STATUS contains information about the error record.

The register indicates whether:

- Any error has been detected.
- Any detected error was not corrected and returned to a master.
- Any detected error was not corrected and deferred.
- A second error of the same type was detected before software handled the first error.
- Any error has been reported.
- The other error record registers contain valid information.

Bit field descriptions

ERR0STATUS is a 32-bit register.

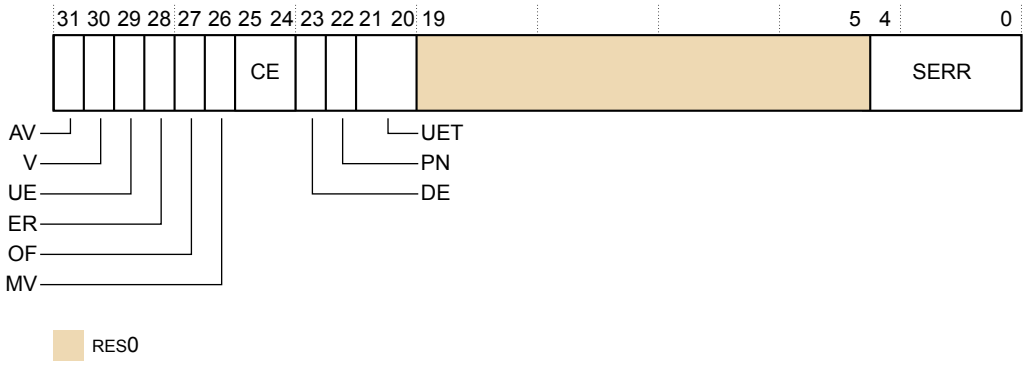

Figure B3-8 ERR0STATUS bit assignments

AV, [31]

Address Valid. The possible values are:

0 ERR0ADDR is not valid.

1 ERR0ADDR contains an address associated with the highest priority error recorded by this record.

V, [30]

Status Register valid. The possible values are:

0 ERR0STATUS is not valid.

1 ERR0STATUS is valid. At least one error has been recorded.

UE, [29]

Uncorrected error. The possible values are:

0 No error that could not be corrected or deferred has been detected.

1 At least one error that could not be corrected or deferred has been detected. If error recovery interrupts are enabled, then the interrupt signal is asserted until this bit is cleared.

ER, [28]

Error reported. The possible values are:

0 No External abort has been reported.

1 The node has reported an External abort to the master that is in access or making a transaction.

OF, [27]

Overflow. The possible values are:

0
- If UE == 1, then no error status for an Uncorrected error has been discarded.
- If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded.
- If UE == 0, DE == 0, and CE !== 0b00, then:

  The corrected error counter has not overflowed.

1 More than one error has occurred and so details of the other error have been discarded.

MV, [26]

Miscellaneous Registers Valid. The possible values are:

0 ERR0MISC0 and ERR0MISC1 are not valid.

1 This bit indicates that ERR0MISC0 contains additional information about any error that is recorded by this record.

CE, [25:24]

Corrected error. The possible values are:

0b00 No corrected error recorded

0b10 At least one corrected error recorded

DE, [23]

Deferred error. The possible values are:

0 No errors deferred

1 At least one error not corrected and deferred by poisoning

PN, [22]

Poison. The value is:

0 The Cortex-A78C core cannot distinguish a poisoned value from a corrupted value.

UET, [21:20]

Uncorrected Error Type. The value is:

0b00 Uncontainable

RES0, [19:5]

RES0 Reserved

SERR, [4:0]

Primary error code. The possible values are:

0x0 No error

0x1 Errors due to fault injection

0x2 ECC error from internal data buffer

0x6 ECC error on cache data RAM

0x7 ECC error on cache tag or dirty RAM

0x8 Parity error on TLB data RAM

0x12 Error response for a cache copyback

0x15 Deferred error from slave not supported at the consumer. For example, poisoned data received from a slave by a master that cannot defer the error further.

Configurations

There are no configuration notes.

ERR0STATUS resets to `0x00000000`.

When ERRSELR_EL1.SEL==0, ERR0STATUS is accessible from *B2.69 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1* on page B2-251

Chapter B4
GIC registers

This chapter describes the GIC registers.

It contains the following sections:

B4.1 CPU interface registers

Each CPU interface block provides the interface for the Cortex-A78C core that interfaces with a GIC distributor within the system.

The Cortex-A78C core only supports System register access to the GIC CPU interface registers. The following table lists the three types of GIC CPU interface System registers supported in the Cortex-A78C core.

Table B4-1 GIC CPU interface system register types supported in the Cortex-A78C core

Register prefix	Register type
ICC	Physical GIC CPU interface System registers
ICV	Virtual GIC CPU interface System registers
ICH	Virtual interface control System registers

Access to virtual GIC CPU interface System registers is only possible at Non-secure EL1.

Access to ICC registers or the equivalent ICV registers is determined by HCR_EL2. See *B2.74 HCR_EL2, Hypervisor Configuration Register, EL2* on page B2-256.

For more information on the CPU interface, see the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.2 AArch64 physical GIC CPU interface system register summary

The following table lists the AArch64 physical GIC CPU interface system registers that have IMPLEMENTATION DEFINED bits.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information and a complete list of AArch64 physical GIC CPU interface system registers.

Table B4-2 AArch64 physical GIC CPU interface system register summary

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICC_AP0R0_EL1	3	0	12	8	4	RW	*B4.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1* on page B4-371
ICC_AP1R0_EL1	3	0	12	9	0	RW	*B4.4 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1* on page B4-372
ICC_BPR0_EL1	3	0	12	8	3	RW	*B4.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1* on page B4-373
ICC_BPR1_EL1	3	0	12	12	3	RW	*B4.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1* on page B4-374
ICC_CTLR_EL1	3	0	12	12	4	RW	*B4.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1* on page B4-375
ICC_CTLR_EL3	3	6	12	12	4	RW	*B4.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3* on page B4-377
ICC_SRE_EL1	3	0	12	12	5	RW	*B4.9 ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1* on page B4-379
ICC_SRE_EL2	3	4	12	9	5	RW	*B4.10 ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2* on page B4-380
ICC_SRE_EL3	3	6	12	12	5	RW	*B4.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3* on page B4-382

B4.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1

The ICC_AP0R0_EL1 provides information about Group 0 active priorities.

Bit descriptions

This register is a 32-bit register and is part of:

- The GIC system registers functional group
- The GIC control registers functional group

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

`0x00000000` No interrupt active. This is the reset value.

`0x00000001` Interrupt active for priority `0x0`

`0x00000002` Interrupt active for priority `0x8`

...

`0x80000000` Interrupt active for priority `0xF8`

Details not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.4 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1

The ICC_AP1R0_EL1 provides information about Group 1 active priorities.

Bit descriptions

This register is a 32-bit register and is part of:

- The GIC system registers functional group
- The GIC control registers functional group

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

`0x00000000` No interrupt active. This is the reset value.

`0x00000001` Interrupt active for priority `0x0`

`0x00000002` Interrupt active for priority `0x8`

...

`0x80000000` Interrupt active for priority `0xF8`

Details not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1

ICC_BPR0_EL1 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.

Bit field descriptions

ICC_BPR0_EL1 is a 32-bit register and is part of:

- The GIC system registers functional group
- The GIC control registers functional group

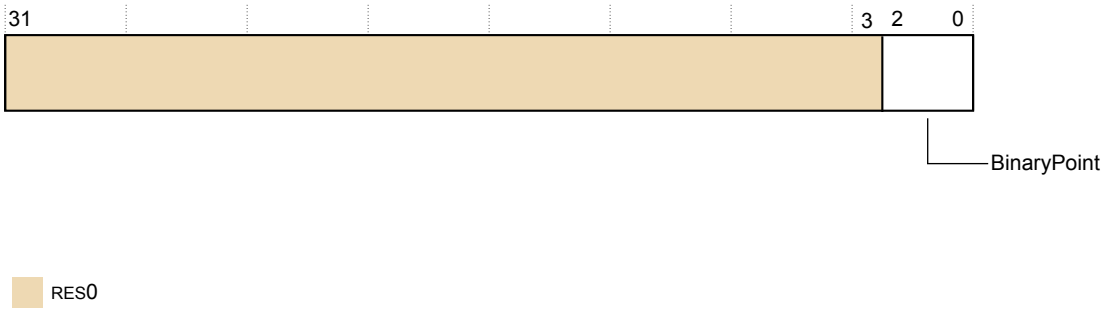

Figure B4-1 ICC_BPR0_EL1 bit assignments

RES0, [31:3]

RES0 Reserved

BinaryPoint, [2:0]

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. The minimum value that is implemented is:

0x2

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1

ICC_BPR1_EL1 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.

Bit field descriptions

ICC_BPR1_EL1 is a 32-bit register and is part of:

- The GIC system registers functional group
- The GIC control registers functional group

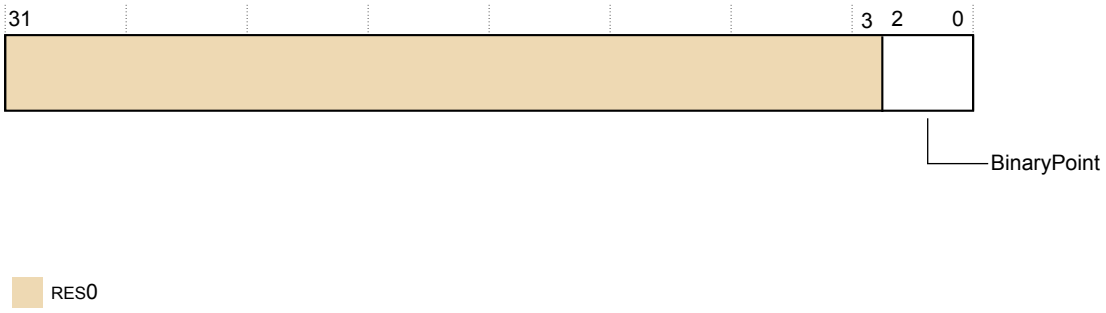

Figure B4-2 ICC_BPR1_EL1 bit assignments

RES0, [31:3]

RES0 Reserved

BinaryPoint, [2:0]

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.

The minimum value implemented of ICC_BPR1_EL1 Secure register is `0x2`.

The minimum value implemented of ICC_BPR1_EL1 Non-secure register is `0x3`.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1

ICC_CTLR_EL1 controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Bit field descriptions

ICC_CTLR_EL1 is a 32-bit register and is part of:

- The GIC System registers functional group
- The GIC control registers functional group

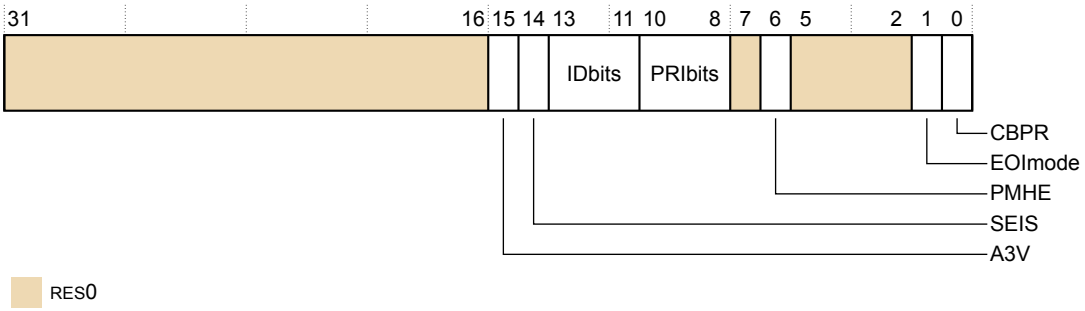

Figure B4-3 ICC_CTLR_EL1 bit assignments

RES0, [31:16]

RES0 Reserved

A3V, [15]

Affinity 3 Valid. The value is:

1 The CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers.

SEIS, [14]

SEI Support. The value is:

0 The CPU interface logic does not support local generation of SEIs.

IDbits, [13:11]

Identifier bits. The value is:

0 The number of physical interrupt identifier bits supported is 16 bits.

This field is an alias of ICC_CTLR_EL3.ID bits.

PRIbits, [10:8]

Priority bits. The value is:

0x4 The core supports 32 levels of physical priority with 5 priority bits.

RES0, [7]

RES0 Reserved

PMHE, [6]

Priority Mask Hint Enable. This bit is read-only and is an alias of ICC_CTLR_EL3.PMHE. The possible values are:

0 Disables use of ICC_PMR as a hint for interrupt distribution.

1 Enables use of ICC_PMR as a hint for interrupt distribution.

RES0, [5:2]

RES0 Reserved

EOImode, [1]

End of interrupt mode for the current Security state. The possible values are:

0 ICC_EOIR0 and ICC_EOIR1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR are UNPREDICTABLE.

1 ICC_EOIR0 and ICC_EOIR1 provide priority drop functionality only. ICC_DIR provides interrupt deactivation functionality.

CBPR, [0]

Common Binary Point Register. Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupt. The possible values are:

0 ICC_BPR0 determines the preemption group for Group 0 interrupts.

ICC_BPR1 determines the preemption group for Group 1 interrupts.

1 ICC_BPR0 determines the preemption group for Group 0 and Group 1 interrupts.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3

ICC_CTLR_EL3 controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Bit field descriptions

ICC_CTLR_EL3 is a 32-bit register and is part of:

- The GIC system registers functional group
- The Security registers functional group
- The GIC control registers functional group

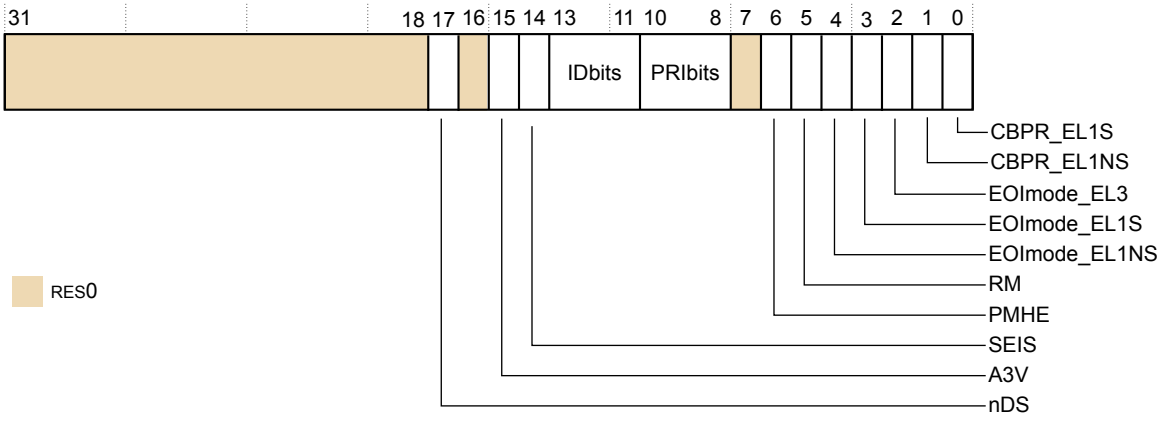

Figure B4-4 ICC_CTLR_EL3 bit assignments

RES0, [31:18]

RES0 Reserved

nDS, [17]

Disable Security not supported. Read-only and writes are IGNORED. The value is:

1 The CPU interface logic does not support disabling of security, and requires that security is not disabled.

RES0, [16]

RES0 Reserved

A3V, [15]

1 Affinity 3 Valid. This bit is RAO/WI.

SEIS, [14]

SEI Support. The value is:

0 The CPU interface logic does not support generation of SEIs.

IDbits, [13:11]

Identifier bits. The value is:

0x0 The number of physical interrupt identifier bits supported is 16 bits.

This field is an alias of ICC_CTLR_EL3.IDbits.

PRIbits, [10:8]

Priority bits. The value is:

`0x4` The core supports 32 levels of physical priority with 5 priority bits.
Accesses to ICC_AP0R{1—3} and ICC_AP1R{1—3} are UNDEFINED.

RES0, [7]

RES0 Reserved

PMHE, [6]

Priority Mask Hint Enable. The possible values are:

0 Disables use of ICC_PMR as a hint for interrupt distribution.

1 Enables use of ICC_PMR as a hint for interrupt distribution.

RM, [5]

Routing Modifier. This bit is RAZ/WI.

EOImode_EL1NS, [4]

EOI mode for interrupts is handled at Non-secure EL1 and EL2.

Controls whether a write to an End of Interrupt register also deactivates the interrupt.

EOImode_EL1S, [3]

EOI mode for interrupts is handled at Secure EL1.

Controls whether a write to an End of Interrupt register also deactivates the interrupt.

EOImode_EL3, [2]

EOI mode for interrupts is handled at EL3.

Controls whether a write to an End of Interrupt register also deactivates the interrupt.

CBPR_EL1NS, [1]

Common Binary Point Register, EL1 Non-secure

Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2.

CBPR_EL1S, [0]

Common Binary Point Register, EL1 Secure

Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupt at EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.9 ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1

ICC_SRE_EL1 controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL0 and EL1.

Bit field descriptions

ICC_SRE_EL1 is a 32-bit register and is part of:

- The GIC system registers functional group
- The GIC control registers functional group

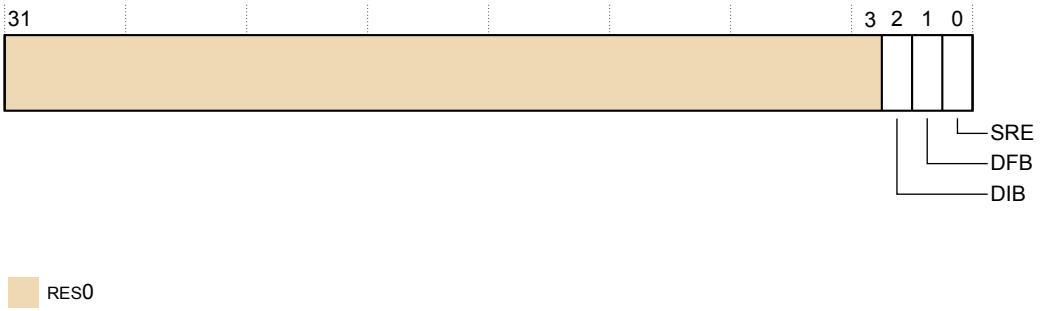

Figure B4-5 ICC_SRE_EL1 bit assignments

RES0, [31:3]

RES0 Reserved

DIB, [2]

Disable IRQ bypass. The possible values are:

0x0 IRQ bypass enabled

0x1 IRQ bypass disabled

This bit is an alias of ICC_SRE_EL3.DIB

DFB, [1]

Disable FIQ bypass. The possible values are:

0x0 FIQ bypass enabled

0x1 FIQ bypass disabled

This bit is an alias of ICC_SRE_EL3.DFB

SRE, [0]

System Register Enable. The value is:

0x1 The System register interface for the current Security state is enabled.

This bit is RAO/WI. The core only supports a system register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.10 ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2

ICC_SRE_EL2 controls whether the system register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

Bit field descriptions

ICC_SRE_EL2 is a 32-bit register and is part of:

- The GIC system registers functional group
- The Virtualization registers functional group
- The GIC control registers functional group

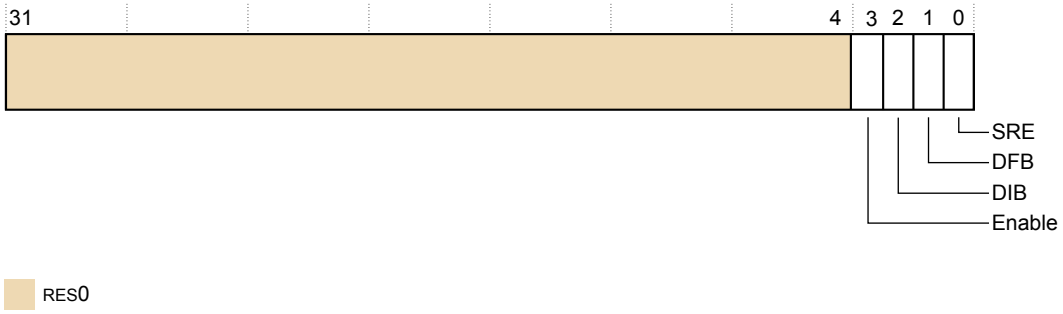

Figure B4-6 ICC_SRE_EL2 bit assignments

RES0, [31:4]

RES0 Reserved

Enable, [3]

Enables lower Exception level access to ICC_SRE_EL1. The value is:

0x1 Non-secure EL1 accesses to ICC_SRE_EL1 do not trap to EL2.

This bit is RAO/WI.

DIB, [2]

Disable IRQ bypass. The possible values are:

0x0 IRQ bypass enabled

0x1 IRQ bypass disabled

This bit is an alias of ICC_SRE_EL3.DIB

DFB, [1]

Disable FIQ bypass. The possible values are:

0x0 FIQ bypass enabled

0x1 FIQ bypass disabled

This bit is an alias of ICC_SRE_EL3.DFB

SRE, [0]

System Register Enable. The value is:

0x1 The System register interface for the current Security state is enabled.

This bit is RAO/WI. The core only supports a system register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3

ICC_SRE_EL3 controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL3.

Bit field descriptions

ICC_SRE_EL3 is a 32-bit register and is part of:

- The GIC system registers functional group
- The Security registers functional group
- The GIC control registers functional group

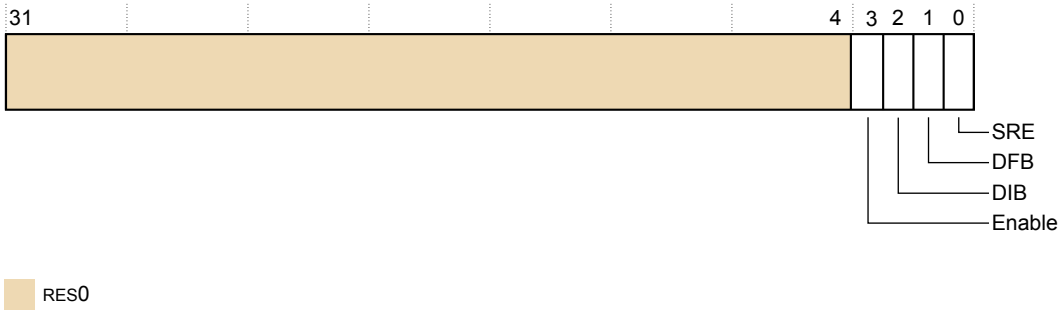

Figure B4-7 ICC_SRE_EL3 bit assignments

RES0, [31:4]

RES0 Reserved

Enable, [3]

Enables lower Exception level access to ICC_SRE_EL1 and ICC_SRE_EL2. The value is:

1
- Secure EL1 accesses to Secure ICC_SRE_EL1 do not trap to EL3.
- EL2 accesses to Non-secure ICC_SRE_EL1 and ICC_SRE_EL2 do not trap to EL3.
- Non-secure EL1 accesses to ICC_SRE_EL1 do not trap to EL3.

This bit is RAO/WI.

DIB, [2]

Disable IRQ bypass. The possible values are:

0 IRQ bypass enabled

1 IRQ bypass disabled

DFB, [1]

Disable FIQ bypass. The possible values are:

0 FIQ bypass enabled

1 FIQ bypass disabled

SRE, [0]

System Register Enable. The value is:

1 The System register interface for the current Security state is enabled.

This bit is RAO/WI. The core only supports a system register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.12 AArch64 virtual GIC CPU interface register summary

The following table describes the AArch64 virtual GIC CPU interface system registers that have IMPLEMENTATION DEFINED bits.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information and a complete list of AArch64 virtual GIC CPU interface system registers.

Table B4-3 AArch64 virtual GIC CPU interface register summary

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICV_AP0R0_EL1	3	0	12	8	4	RW	*B4.13 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1* on page B4-385
ICV_AP1R0_EL1	3	0	12	9	0	RW	*B4.14 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1* on page B4-386
ICV_BPR0_EL1	3	0	12	8	3	RW	*B4.15 ICV_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1* on page B4-387
ICV_BPR1_EL1	3	0	12	12	3	RW	*B4.16 ICV_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1* on page B4-388
ICV_CTLR_EL1	3	0	12	12	4	RW	*B4.17 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1* on page B4-389

B4.13 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1

The ICV_AP0R0_EL1 register provides information about virtual Group 0 active priorities.

Bit descriptions

This register is a 32-bit register and is part of the virtual GIC system registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

`0x00000000` No interrupt active. This is the reset value.

`0x00000001` Interrupt active for priority `0x0`

`0x00000002` Interrupt active for priority `0x8`

...

`0x80000000` Interrupt active for priority `0xF8`

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.14 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1

The ICV_AP1R0_EL1 register provides information about virtual Group 1 active priorities.

Bit descriptions

This register is a 32-bit register and is part of the virtual GIC system registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

`0x00000000` No interrupt active. This is the reset value.

`0x00000001` Interrupt active for priority `0x0`

`0x00000002` Interrupt active for priority `0x8`

...

`0x80000000` Interrupt active for priority `0xF8`

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.15 ICV_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1

ICV_BPR0_EL1 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.

Bit field descriptions

ICC_BPR0_EL1 is a 32-bit register and is part of the virtual GIC system registers functional group.

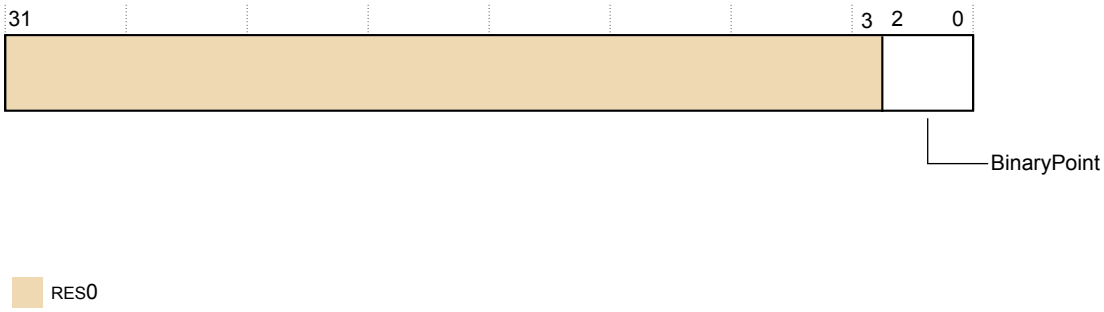

Figure B4-8 ICV_BPR0_EL1 bit assignments

RES0, [31:3]

RES0 Reserved

BinaryPoint, [2:0]

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. The minimum value that is implemented is:

0x2

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.16 ICV_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1

ICV_BPR1_EL1 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.

Bit field descriptions

ICV_BPR1_EL1 is a 32-bit register and is part of the virtual GIC system registers functional group.

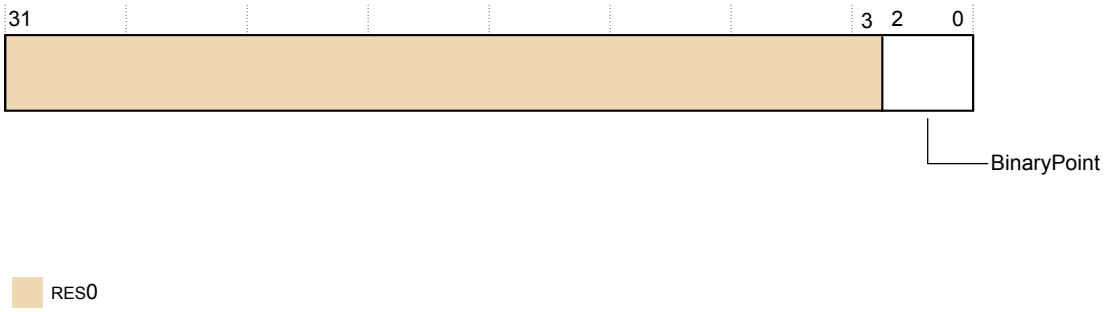

Figure B4-9 ICV_BPR1_EL1 bit assignments

RES0, [31:3]

RES0 Reserved

BinaryPoint, [2:0]

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.

The minimum value that is implemented of ICV_BPR1_EL1 Secure register is `0x2`.

The minimum value that is implemented of ICV_BPR1_EL1 Non-secure register is `0x3`.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.17 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1

ICV_CTLR_EL1 controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Bit field descriptions

ICV_CTLR_EL1 is a 32-bit register and is part of the virtual GIC system registers functional group.

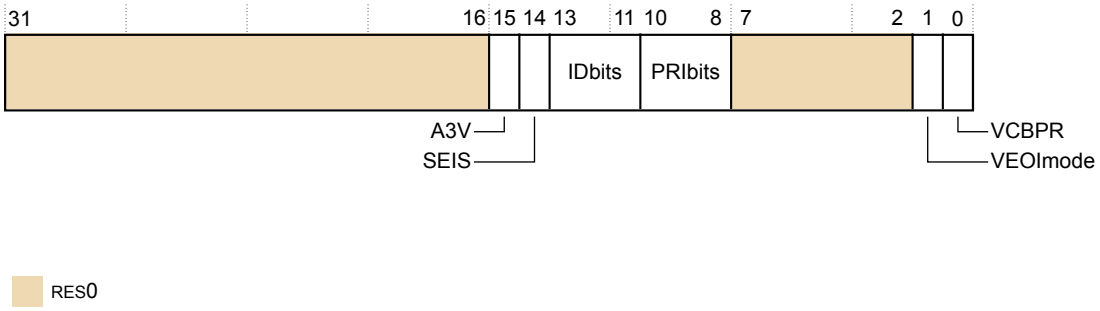

Figure B4-10 ICV_CTLR_EL1 bit assignments

RES0, [31:16]

RES0 Reserved

A3V, [15]

Affinity 3 Valid. The value is:

`0x1` The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

SEIS, [14]

SEI Support. The value is:

`0x0` The virtual CPU interface logic does not support local generation of SEIs.

IDbits, [13:11]

Identifier bits. The value is:

`0x0` The number of physical interrupt identifier bits supported is 16 bits.

PRIbits, [10:8]

Priority bits. The value is:

`0x4` Support 32 levels of physical priority (5 priority bits).

RES0, [7:2]

RES0 Reserved

VEOImode, [1]

Virtual EOI mode. The possible values are:

`0x0` ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV_DIR_EL1 are UNPREDICTABLE.

`0x1` ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide priority drop functionality only. ICV_DIR provides interrupt deactivation functionality.

VCBPR, [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts. The possible values are:

0 ICV_BPR0_EL1 determines the preemption group for virtual Group 0 interrupts only.

ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.

1 ICV_BPR0_EL1 determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts.

Reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 plus one, saturated to 111. Writes to ICV_BPR1_EL1 are IGNORED.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.18 AArch64 virtual interface control system register summary

The following table lists the AArch64 virtual interface control system registers that have IMPLEMENTATION DEFINED bits.

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information and a complete list of AArch64 virtual interface control system registers.

Table B4-4 AArch64 virtual interface control system register summary

Name	Op0	Op1	CRn	CRm	Op2	Type	Description
ICH_AP0R0_EL2	3	0	12	8	4	RW	*B4.19 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2* on page B4-392
ICH_AP1R0_EL2	3	0	19	9	0	RW	*B4.20 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2* on page B4-393
ICH_HCR_EL2	3	4	12	11	0	RW	*B4.21 ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2* on page B4-394
ICH_VTR_EL2	3	4	12	11	1	RO	*B4.22 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2* on page B4-397
ICH_VMCR_EL2	3	4	12	11	7	RW	*B4.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2* on page B4-399

B4.19 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2

The ICH_AP0R0_EL2 provides information about Group 0 active priorities for EL2.

Bit field descriptions

This register is a 32-bit register and is part of:

- The GIC system registers functional group
- The Virtualization registers functional group
- The GIC host interface control registers functional group

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

`0x00000000` No interrupt active. This is the reset value.

`0x00000001` Interrupt active for priority `0x0`

`0x00000002` Interrupt active for priority `0x8`

...

`0x80000000` Interrupt active for priority `0xF8`

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.20 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2

The ICH_AP1R0_EL2 provides information about Group 1 active priorities for EL2.

Bit field descriptions

This register is a 32-bit register and is part of:

- The GIC system registers functional group
- The Virtualization registers functional group
- The GIC host interface control registers functional group

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

`0x00000000` No interrupt active. This is the reset value.

`0x00000001` Interrupt active for priority `0x0`

`0x00000002` Interrupt active for priority `0x8`

...

`0x80000000` Interrupt active for priority `0xF8`

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.21 ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2

ICH_HCR_EL2 controls the environment for VMs.

Bit field descriptions

ICH_HCR_EL2 is a 32-bit register and is part of:

- The GIC system registers functional group
- The Virtualization registers functional group
- The GIC host interface control registers functional group

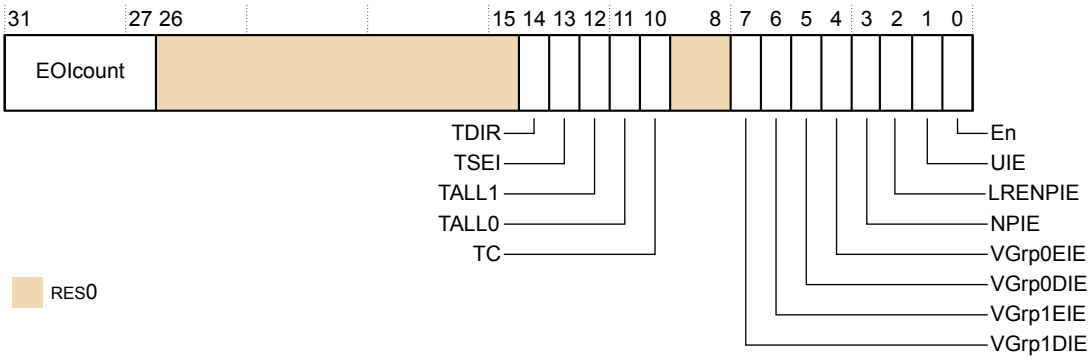

Figure B4-11 ICH_HCR_EL2 bit assignments

EOIcount, [31:27]

Number of outstanding deactivates

RES0, [26:15]

RES0 Reserved

TDIR, [14]

Trap Non-secure EL1 writes to ICC_DIR_EL1 and ICV_DIR_EL1. The possible values are:

`0x0` Non-secure EL1 writes of ICC_DIR_EL1 and ICV_DIR_EL1 are not trapped to EL2, unless trapped by other mechanisms.

`0x1` Non-secure EL1 writes of ICC_DIR_EL1 and ICV_DIR_EL1 are trapped to EL2.

TSEI, [13]

Trap all locally generated SEIs. The value is:

0 Locally generated SEIs do not cause a trap to EL2.

TALL1, [12]

Trap all Non-secure EL1 accesses to ICC_* and ICV_* System registers for Group 1 interrupts to EL2. The possible values are:

`0x0` Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts proceed as normal.

`0x1` Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts trap to EL2.

TALL0, [11]

Trap all Non-secure EL1 accesses to ICC_* and ICV_* System registers for Group 0 interrupts to EL2. The possible values are:

`0x0` Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts proceed as normal.

`0x1` Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts trap to EL2.

TC, [10]

Trap all Non-secure EL1 accesses to System registers that are common to Group 0 and Group 1 to EL2. The possible values are:

`0x0` Non-secure EL1 accesses to common registers proceed as normal.

`0x1` Non-secure EL1 accesses to common registers trap to EL2.

RES0, [9:8]

RES0 Reserved

VGrp1DIE, [7]

VM Group 1 Disabled Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled

1 Maintenance interrupt signaled when ICH_VMCR_EL2.VENG1 is 0.

VGrp1EIE, [6]

VM Group 1 Enabled Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled

1 Maintenance interrupt signaled when ICH_VMCR_EL2.VENG1 is 1.

VGrp0DIE, [5]

VM Group 0 Disabled Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled

1 Maintenance interrupt signaled when ICH_VMCR_EL2.VENG0 is 0.

VGrp0EIE, [4]

VM Group 0 Enabled Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled

1 Maintenance interrupt signaled when ICH_VMCR_EL2.VENG0 is 1.

NPIE, [3]

No Pending Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled

1 Maintenance interrupt signaled while the List registers contain no interrupts in the pending state.

LRENPIE, [2]

List Register Entry Not Present Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled

1 Maintenance interrupt is asserted while the EOIcount field is not 0.

UIE, [1]

Underflow Interrupt Enable. The possible values are:

0 Maintenance interrupt disabled

1 Maintenance interrupt is asserted if none, or only one, of the List register entries is marked as a valid interrupt.

En, [0]

Enable. The possible values are:

0 Virtual CPU interface operation disabled

1 Virtual CPU interface operation enabled

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.22 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2

ICH_VMCR_EL2 enables the hypervisor to save and restore the virtual machine view of the GIC state.

Bit field descriptions

ICH_VMCR_EL2 is a 32-bit register and is part of:

- The GIC system registers functional group
- The Virtualization registers functional group
- The GIC host interface control registers functional group

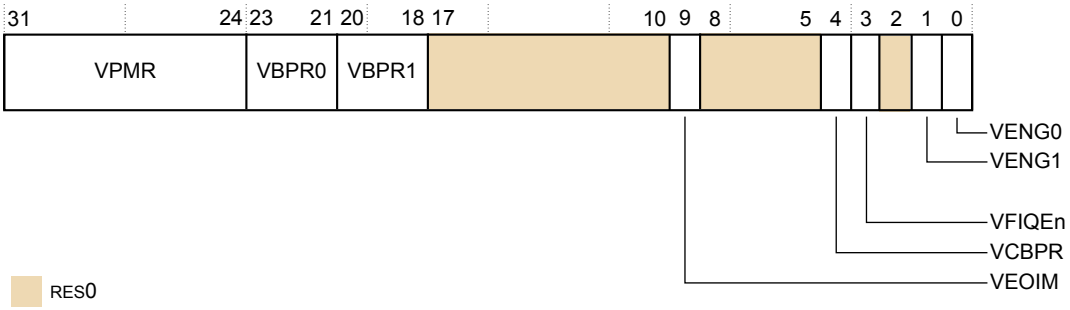

Figure B4-12 ICH_VMCR_EL2 bit assignments

VPMR, [31:24]

Virtual Priority Mask

This field is an alias of ICV_PMR_EL1.Priority.

VBPR0, [23:21]

Virtual Binary Point Register, Group 0. The minimum value is:

`0x2` This field is an alias of ICV_BPR0_EL1.BinaryPoint.

VBPR1, [20:18]

Virtual Binary Point Register, Group 1. The minimum value is:

`0x3` This field is an alias of ICV_BPR1_EL1.BinaryPoint.

RES0, [17:10]

RES0 Reserved

VEOIM, [9]

Virtual EOI mode. The possible values are:

`0x0` ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV_DIR_EL1 are UNPREDICTABLE.

`0x1` ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide priority drop functionality only. ICV_DIR_EL1 provides interrupt deactivation functionality.

This bit is an alias of ICV_CTLR_EL1.EOImode.

RES0, [8:5]

RES0 Reserved

VCBPR, [4]

Virtual Common Binary Point Register. The possible values are:

`0x0` ICV_BPR0_EL1 determines the preemption group for virtual Group 0 interrupts only.

ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.

`0x1` ICV_BPR0_EL1 determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts.

Reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 plus one, saturated to 111. Writes to ICV_BPR1_EL1 are IGNORED.

VFIQEn, [3]

Virtual FIQ enable. The value is:

`0x1` Group 0 virtual interrupts are presented as virtual FIQs.

RES0, [2]

RES0 Reserved

VENG1, [1]

Virtual Group 1 interrupt enable. The possible values are:

`0x0` Virtual Group 1 interrupts are disabled.

`0x1` Virtual Group 1 interrupts are enabled.

VENG0, [0]

Virtual Group 0 interrupt enable. The possible values are:

`0x0` Virtual Group 0 interrupts are disabled.

`0x1` Virtual Group 0 interrupts are enabled.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

B4.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2

ICH_VTR_EL2 reports supported GIC virtualization features.

Bit field descriptions

ICH_VTR_EL2 is a 32-bit register and is part of:

- The GIC system registers functional group
- The Virtualization registers functional group
- The GIC host interface control registers functional group

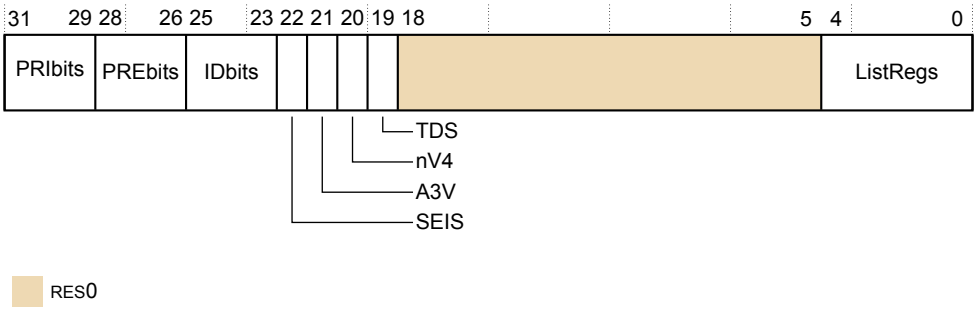

Figure B4-13 ICH_VTR_EL2 bit assignments

PRIbits, [31:29]

Priority bits. The number of virtual priority bits implemented, minus one.

`0x4` Priority implemented is 5-bit.

PREbits, [28:26]

The number of virtual preemption bits implemented, minus one. The value is:

`0x4` Virtual preemption implemented is 5-bit.

IDbits, [25:23]

The number of virtual interrupt identifier bits supported. The value is:

`0x0` Virtual interrupt identifier bits that are implemented is 16-bit.

SEIS, [22]

SEI Support. The value is:

`0x0` The virtual CPU interface logic does not support generation of SEIs.

A3V, [21]

Affinity 3 Valid. The value is:

`0x1` The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

nV4, [20]

Direct injection of virtual interrupts not supported. The value is:

`0x0` The CPU interface logic supports direct injection of virtual interrupts.

TDS, [19]

Separate trapping of Non-secure EL1 writes to ICV_DIR_EL1 supported. The value is:

`0x1` Implementation supports ICH_HCR_EL2.TDIR.

RES0, [18:5]

RES0 Reserved

ListRegs, [4:0]

`0x3` The number of implemented List registers, minus one.

The core implements 4 list registers. Accesses to ICH_LR_EL2[x] (x>3) in AArch64 are UNDEFINED.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4*.

Chapter B5
Advanced SIMD and floating-point registers

This chapter describes the Advanced SIMD and floating-point registers.

It contains the following sections:

B5.1 AArch64 register summary

The core has several Advanced SIMD and floating-point system registers in the AArch64 Execution state. Each register has a specific purpose, specific usage constraints, configurations, and attributes.

The following table gives a summary of the Cortex-A78C core Advanced SIMD and floating-point system registers in the AArch64 Execution state.

Table B5-1 AArch64 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPCR	RW	0x00000000	*B5.2 FPCR, Floating-point Control Register* on page B5-403
FPSR	RW	UNKNOWN	*B5.3 FPSR, Floating-point Status Register* on page B5-405
MVFR0_EL1	RO	0x10110222	*B5.4 MVFR0_EL1, Media and VFP Feature Register 0, EL1* on page B5-407
MVFR1_EL1	RO	0x13211111	*B5.5 MVFR1_EL1, Media and VFP Feature Register 1, EL1* on page B5-409
MVFR2_EL1	RO	0x00000043	*B5.6 MVFR2_EL1, Media and VFP Feature Register 2, EL1* on page B5-411

B5.2 FPCR, Floating-point Control Register

The FPCR controls floating-point behavior.

Bit field descriptions

FPCR is a 32-bit register.

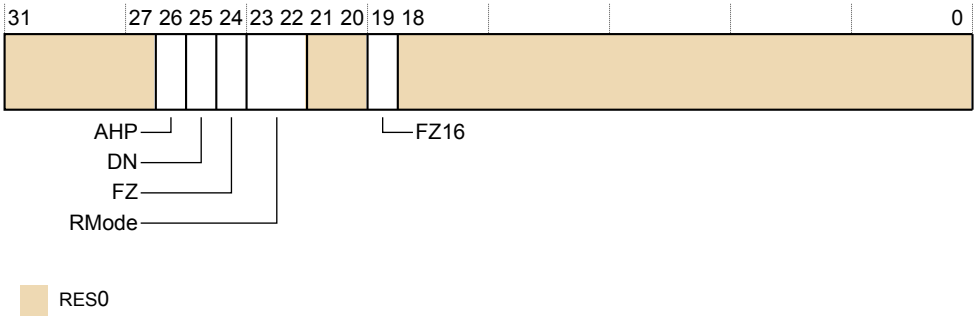

Figure B5-1 FPCR bit assignments

RES0, [31:27]

RES0 Reserved

AHP, [26]

Alternative half-precision control bit. The possible values are:

`0` IEEE half-precision format selected. This is the reset value.

`1` Alternative half-precision format selected

DN, [25]

Default NaN mode control bit. The possible values are:

`0` NaN operands propagate through to the output of a floating-point operation. This is the reset value.

`1` Any operation involving one or more NaNs returns the Default NaN.

FZ, [24]

Flush-to-zero mode control bit. The possible values are:

`0` Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.

`1` Flush-to-zero mode enabled

RMode, [23:22]

Rounding Mode control field. The encoding of this field is:

`0b00` *Round to Nearest* (RN) mode. This is the reset value.

`0b01` *Round towards Plus Infinity* (RP) mode

`0b10` *Round towards Minus Infinity* (RM) mode

`0b11` *Round towards Zero* (RZ) mode

RES0, [21:20]

RES0 Reserved

FZ16, [19]

Flush-to-zero mode control bit on half-precision data-processing instructions. The possible values are:

`0` Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the default value.

`1` Flush-to-zero mode enabled

RES0, [18:0]

RES0 Reserved

Configurations

The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *B5.8 FPSCR, Floating-Point Status and Control Register* on page B5-414.

Usage constraints

Accessing the FPCR

To access the FPCR:

```
MRS <Xt>, FPCR ; Read FPCR into Xt
MSR FPCR, <Xt> ; Write Xt to FPCR
```

Register access is encoded as follows:

Table B5-2 FPCR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	000

Accessibility

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

B5.3 FPSR, Floating-point Status Register

The FPSR provides floating-point system status information.

Bit field descriptions

FPSR is a 32-bit register.

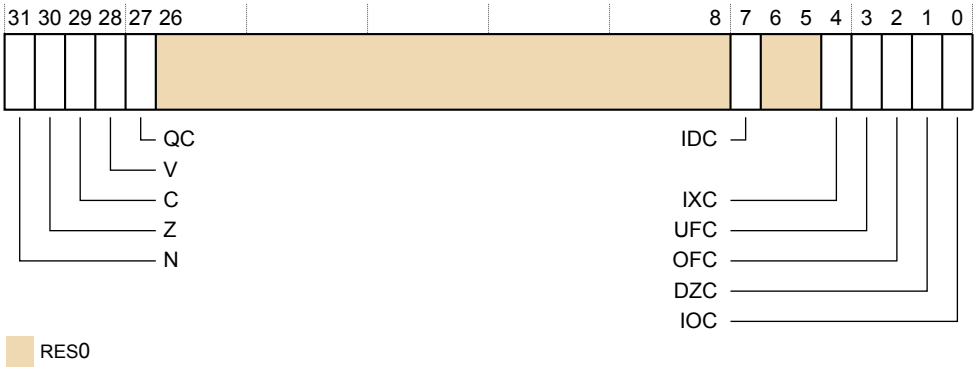

Figure B5-2 FPSR bit assignments

N, [31]

Negative condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.N flag instead.

Z, [30]

Zero condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.Z flag instead.

C, [29]

Carry condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.C flag instead.

V, [28]

Overflow condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.V flag instead.

QC, [27]

Cumulative saturation bit. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since a 0 was last written to this bit.

RES0, [26:8]

RES0 Reserved

IDC, [7]

Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

RES0, [6:5]

RES0 Reserved

IXC, [4]

Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

UFC, [3]

Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.

OFC, [2]

Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

DZC, [1]

Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

IOC, [0]

Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

Configurations

The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *B5.8 FPSCR, Floating-Point Status and Control Register* on page B5-414.

Usage constraints

Accessing the FPSR

To access the FPSR:

```
MRS <Xt>, FPSR; Read FPSR into Xt
MSR FPSR, <Xt>; Write Xt to FPSR
```

Register access is encoded as follows:

Table B5-3 FPSR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	001

Accessibility

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

B5.4 MVFR0_EL1, Media and VFP Feature Register 0, EL1

The MVFR0_EL1 describes the features provided by the AArch64 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR0_EL1 is a 32-bit register.

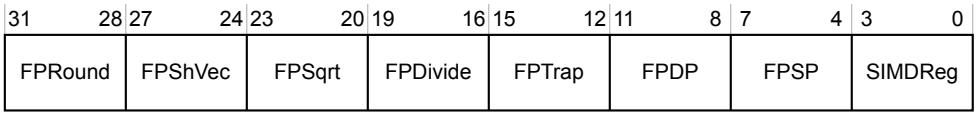

Figure B5-3 MVFR0_EL1 bit assignments

FPRound, [31:28]

Indicates the rounding modes supported by the floating-point hardware:

`0x1` All rounding modes supported

FPShVec, [27:24]

Indicates the hardware support for floating-point short vectors:

`0x0` Not supported

FPSqrt, [23:20]

Indicates the hardware support for floating-point square root operations:

`0x1` Supported

FPDivide, [19:16]

Indicates the hardware support for floating-point divide operations:

`0x1` Supported

FPTrap, [15:12]

Indicates whether the floating-point hardware implementation supports exception trapping:

`0x0` Not supported

FPDP, [11:8]

Indicates the hardware support for floating-point double-precision operations:

`0x2` Supported, VFPv3 or greater

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

FPSP, [7:4]

Indicates the hardware support for floating-point single-precision operations:

`0x2` Supported, VFPv3 or greater

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

SIMDReg, [3:0]

Indicates support for the Advanced SIMD register bank:

`0x2` Supported, 32 x 64-bit registers supported

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

Configurations

There are no configuration notes.

Usage constraints

Accessing the MVFR0_EL1

To access the MVFR0_EL1:

```
MRS <Xt>, MVFR0_EL1 ; Read MVFR0_EL1 into Xt
```

Register access is encoded as follows:

Table B5-4 MVFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	000

Accessibility

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

B5.5 MVFR1_EL1, Media and VFP Feature Register 1, EL1

The MVFR1_EL1 describes the features provided by the AArch64 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR1_EL1 is a 32-bit register.

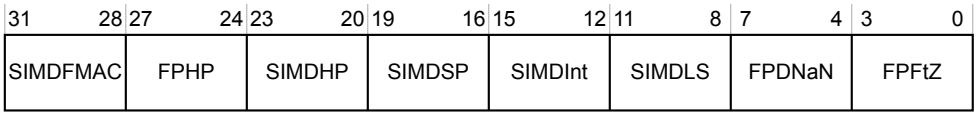

Figure B5-4 MVFR1_EL1 bit assignments

SIMDFMAC, [31:28]

Indicates whether the Advanced SIMD and floating-point unit supports fused multiply accumulate operations:

1 Implemented

FPHP, [27:24]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion instructions:

3 Floating-point half-precision conversion and data processing instructions implemented

SIMDHP, [23:20]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion operations:

2 Advanced SIMD half-precision conversion and data processing instructions implemented

SIMDSP, [19:16]

Indicates whether the Advanced SIMD and floating-point unit supports single-precision floating-point operations:

1 Implemented

SIMDInt, [15:12]

Indicates whether the Advanced SIMD and floating-point unit supports integer operations:

1 Implemented

SIMDLS, [11:8]

Indicates whether the Advanced SIMD and floating-point unit supports load/store instructions:

1 Implemented

FPDNaN, [7:4]

Indicates whether the floating-point hardware implementation supports only the Default NaN mode:

1 Hardware supports propagation of NaN values.

FPFtZ, [3:0]

Indicates whether the floating-point hardware implementation supports only the Flush-to-zero mode of operation:

1 Hardware supports full denormalized number arithmetic.

Configurations

There are no configuration notes.

Usage constraints

Accessing the MVFR1_EL1

To access the MVFR1_EL1:

```
MRS <Xt>, MVFR1_EL1 ; Read MVFR1_EL1 into Xt
```

Register access is encoded as follows:

Table B5-5 MVFR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	001

Accessibility

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

B5.6 MVFR2_EL1, Media and VFP Feature Register 2, EL1

The MVFR2_EL1 describes the features provided by the AArch64 Advanced SIMD and floating-point implementation.

Bit field descriptions

MVFR2_EL1 is a 32-bit register.

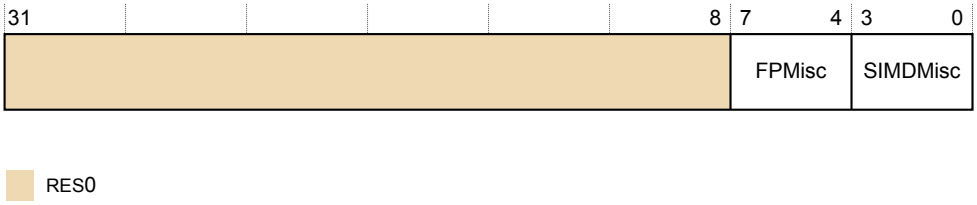

Figure B5-5 MVFR2_EL1 bit assignments

RES0, [31:8]

RES0 Reserved

FPMisc, [7:4]

Indicates support for miscellaneous floating-point features.

0x4 Supports:

- Floating-point selection
- Floating-point Conversion to Integer with Directed Rounding modes
- Floating-point Round to Integral Floating-point
- Floating-point MaxNum and MinNum

SIMDMisc, [3:0]

Indicates support for miscellaneous Advanced SIMD features.

0x3 Supports:

- Floating-point Conversion to Integer with Directed Rounding modes
- Floating-point Round to Integral Floating-point
- Floating-point MaxNum and MinNum

Configurations

There are no configuration notes.

Usage constraints

Accessing the MVFR2_EL1

To access the MVFR2_EL1:

```
MRS <Xt>, MVFR2_EL1 ; Read MVFR2_EL1 into Xt
```

Register access is encoded as follows:

Table B5-6 MVFR2_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	010

Accessibility

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

B5.7 AArch32 register summary

The core has one Advanced SIMD and floating-point System registers in the AArch32 Execution state.

The following table gives a summary of the Cortex-A78C core Advanced SIMD and floating-point System registers in the AArch32 Execution state.

Table B5-7 AArch32 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPSCR	RW	UNKNOWN	*B5.8 FPSCR, Floating-Point Status and Control Register* on page B5-414

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on permitted accesses to the Advanced SIMD and floating-point System registers.

B5.8 FPSCR, Floating-Point Status and Control Register

The FPSCR provides floating-point system status information and control.

Bit field descriptions

FPSCR is a 32-bit register.

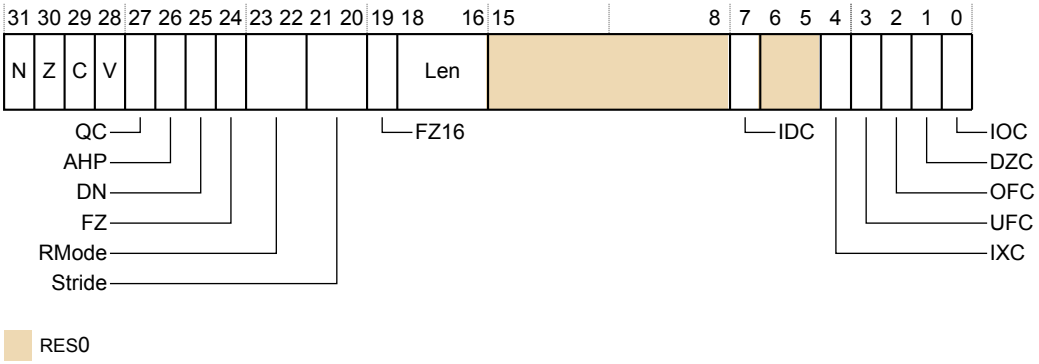

Figure B5-6 FPSCR bit assignments

N, [31]

Floating-point Negative condition code flag

Set to 1 if a floating-point comparison operation produces a less than result.

Z, [30]

Floating-point Zero condition code flag

Set to 1 if a floating-point comparison operation produces an equal result.

C, [29]

Floating-point Carry condition code flag

Set to 1 if a floating-point comparison operation produces an equal, greater than, or unordered result.

V, [28]

Floating-point Overflow condition code flag

Set to 1 if a floating-point comparison operation produces an unordered result.

QC, [27]

Cumulative saturation bit

This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated after 0 was last written to this bit.

AHP, [26]

Alternative Half-Precision control bit:

`0` IEEE half-precision format selected. This is the reset value.

`1` Alternative half-precision format selected

DN, [25]

Default NaN mode control bit:

`0` NaN operands propagate through to the output of a floating-point operation. This is the reset value.

`1` Any operation involving one or more NaNs returns the Default NaN.

The value of this bit only controls floating-point arithmetic. AArch32 Advanced SIMD arithmetic always uses the Default NaN setting, regardless of the value of the DN bit.

FZ, [24]

Flush-to-zero mode control bit:

`0` Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.

`1` Flush-to-zero mode enabled.

The value of this bit only controls floating-point arithmetic. AArch32 Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit.

RMode, [23:22]

Rounding Mode control field:

`0b00` *Round to Nearest* (RN) mode. This is the reset value.

`0b01` *Round towards Plus Infinity* (RP) mode

`0b10` *Round towards Minus Infinity* (RM) mode

`0b11` *Round towards Zero* (RZ) mode

The specified rounding mode is used by almost all floating-point instructions. AArch32 Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.

Stride, [21:20]

RES0 Reserved

FZ16, [19]

Flush-to-zero mode control bit on half-precision data-processing instructions:

`0` Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.

`1` Flush-to-zero mode enabled

Len, [18:16]

RES0 Reserved

RES0, [15:8]

RES0 Reserved

IDC, [7]

Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

RES0, [6:5]

RES0 Reserved

IXC, [4]

Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

UFC, [3]

Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.

OFC, [2]

Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

DZC, [1]

Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

IOC, [0]

Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

The named fields in this register map to the equivalent fields in the AArch64 FPCR and FPSR. See *B5.2 FPCR, Floating-point Control Register* on page B5-403 and *B5.3 FPSR, Floating-point Status Register* on page B5-405.

Usage constraints

Accessing the FPSCR

To access the FPSCR:

```
VMRS <Rt>, FPSCR ; Read FPSCR into Rt
VMSR FPSCR, <Rt> ; Write Rt to FPSCR
```

Register access is encoded as follows:

Table B5-8 FPSCR access encoding

spec_reg
0001

Note

The Cortex-A78C core implementation does not support the deprecated VFP short vector feature. Attempts to execute the associated VFP data-processing instructions result in an UNDEFINED Instruction exception.

Accessibility

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
Config	RW	-	-	-	-	-

Access to this register depends on the values of CPACR_EL1.FPEN, CPTR_EL2.FPEN, CPTR_EL2.TFP, CPTR_EL3.TFP, and HCR_EL2.{E2H, TGE}. For details of which values of these fields allow access at which Exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Part C
Debug descriptions

Chapter C1
Debug

This chapter describes the Cortex-A78C core debug registers and shows examples of how to use them.

It contains the following sections:

C1.1 About debug methods

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

The following figure shows a typical external debug system.

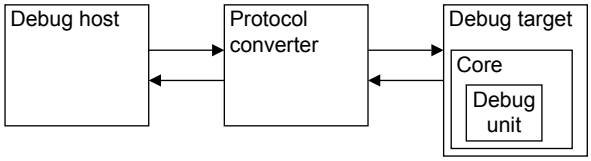

Figure C1-1 External debug system

Debug host

A computer, for example a personal computer, that is running a software debugger. With the debug host, you can issue high-level commands, such as setting a breakpoint at a certain location or examining the contents of a memory address.

Protocol converter

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

Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit using the *Advanced Peripheral Bus* (APB) slave interface. An example of a debug target is a development system with a test chip or a silicon part with a core.

Debug unit

Helps debugging software that is running on the core:

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

With the debug unit, you can:

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

For self-hosted debug, the debug target runs additional debug monitor software that runs on the Cortex-A78C core itself. This way, it does not require expensive interface hardware to connect a second host computer.

C1.2 Debug register interfaces

The Debug architecture defines a set of Debug registers.

The Debug register interfaces provide access to these registers from:

- Software running on the core
- An external debugger

The Cortex-A78C core implements the Armv8 Debug architecture and debug events as described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. It also implements improvements to Debug introduced in Armv8.1 and Armv8.2.

C1.2.1 Core interfaces

System register access allows the core to directly access certain debug registers.

Debug registers
This function is system register based and memory-mapped. You can access the debug register map using the APB slave port. The external debug interface enables both external and self-hosted debug agents to access debug registers.

Performance monitor
This function is system register based and memory-mapped. You can access the performance monitor registers using the APB slave port.

Activity monitor
This function is system register based and memory-mapped. You can access the activity monitor registers using the APB slave port.

Trace registers
This function is memory-mapped.

ELA registers
This function is memory-mapped.

Related references
C1.4 External debug interface on page C1-424

C1.2.2 Breakpoints and watchpoints

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

A breakpoint consists of a breakpoint control register and a breakpoint value register. These two registers are referred to as a *Breakpoint Register Pair* (BRP).

Four of the breakpoints (BRP 0 - BRP 3) match only to virtual address and the other two breakpoints (BRP 4 - BRP 5) match against either virtual address or context ID, or VMID. All the watchpoints can be linked to two breakpoints (BRP 4 - BRP 5) to enable a memory request to be trapped in a given process context.

C1.2.3 Effects of resets on debug registers

The core has the following reset signals that affect the Debug registers:

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

nCORERESET
This signal resets some of the debug and performance monitor logic. This maps to a Warm reset that covers reset of the core processing logic, with the exception of debug.

C1.2.4 External access permissions to debug registers

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

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

Table C1-1 External access conditions to registers

Name	Condition	Description
Off	EDPRSR.PU is 0	Core power domain is completely off, or in a low-power state where the core power domain registers cannot be accessed. If debug power is off, then all external debug and memory-mapped register accesses return an error.
DLK	`DoubleLockStatus() == TRUE` (EDPRSR.DLK is 1)	OS Double Lock is locked.
OSLK	OSLSR_EL1.OSLK is 1	OS Lock is locked.
EDAD	`AllowExternalDebugAccess() ==FALSE`	External debug access is disabled. 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, normal access.

The following table shows an example of external register access condition codes for access to a performance monitor register. To determine the access permission for the register, scan the columns from left to right. Stop at the first column a condition is true, the entry gives the access permission of the register and scanning stops.

Table C1-2 External register condition code example

Off	DLK	OSLK	EDAD	Default
-	-	-	-	RO

C1.3 Debug events

A debug event can be a software debug event or a halting debug event.

A core responds to a debug event in one of the following ways:

- Ignores the debug event
- Takes a debug exception
- Enters debug state

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

C1.3.1 Watchpoint debug events

In the Cortex-A78C core, watchpoint debug events are always synchronous.

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

C1.3.2 Debug OS Lock

Debug OS Lock is set by the powerup reset, **nCPUPORESET**.

For normal behavior of debug events and debug register accesses, Debug OS Lock must be cleared. For more information, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Related references

A3.1 About clocks, resets, and input synchronization on page A3-46
C1.4 External debug interface on page C1-424

C1.4 External debug interface

For information on external debug interface, including debug memory map and debug signals, see the Debug registers and Signal descriptions in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

Chapter C2
Performance Monitoring Unit

This chapter describes the *Performance Monitoring Unit* (PMU) and the registers that it uses.

It contains the following sections:

C2.1 About the PMU

The Cortex-A78C core includes performance monitors that enable you to gather various statistics on the operation of the core and its memory system during runtime. These provide useful information about the behavior of the core that you can use when debugging or profiling code.

The *Performance Monitoring Unit* (PMU) provides six counters. Each counter can count any of the events available in the core. The absolute counts recorded might vary because of pipeline effects. This has negligible effect except in cases where the counters are enabled for a very short time.

Related references

C2.3 PMU events on page C2-428

C2.2 PMU functional description

This section describes the functionality of the *Performance Monitoring Unit* (PMU).

The PMU includes the following interfaces and counters:

Event interface
Events from all other units from across the design are provided to the PMU.

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

Counters
The PMU has 32-bit counters that increment when they are enabled, based on events, and a 64-bit cycle counter.

PMU register interfaces
The Cortex-A78C core supports access to the performance monitor registers from the internal System register interface and a memory-mapped interface.

C2.2.1 External register access permissions

Whether or not access is permitted to a register depends on:

- If the core is powered up
- The state of the OS Lock and OS Double Lock
- The state of External Performance Monitors access disable
- The state of the debug authentication inputs to the core

The behavior is specific to each register and 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 Arm®v8, for Arm®v8-A architecture profile*.

The register descriptions provided in this manual describe whether each register is read/write or read-only.

C2.3 PMU events

The following table shows the events that are generated and the numbers that the *Performance Monitoring Unit* (PMU) uses to reference the events. The table also shows the bit position of each event on the event bus. Event reference numbers that are not listed are reserved.

Table C2-1 PMU Events

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x0	[00]	SW_INCR	Software increment. Instruction architecturally executed (condition code check pass).
0x1	[01]	L1I_CACHE_REFILL	L1 instruction cache refill. This event counts any instruction fetch which misses in the cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses
0x2	[02]	L1I_TLB_REFILL	L1 instruction TLB refill. This event counts any refill of the instruction L1 TLB from the L2 TLB. This includes refills that result in a translation fault. The following instructions are not counted: • TLB maintenance instructions This event counts regardless of whether the MMU is enabled.
0x3	[03]	L1D_CACHE_REFILL	L1 data cache refill. This event counts any load or store operation or page table walk access which causes data to be read from outside the L1, including accesses which do not allocate into L1. The following instructions are not counted: • Cache maintenance instructions and prefetches • Stores of an entire cache line, even if they make a coherency request outside the L1 • Partial cache line writes which do not allocate into the L1 cache • Non-cacheable accesses. This event counts the sum of L1D_CACHE_REFILL_RD and L1D_CACHE_REFILL_WR.
0x4	[06:04]	L1D_CACHE	L1 data cache access. This event counts any load or store operation or page table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x5	[08:07]	L1D_TLB_REFILL	L1 data TLB refill. This event counts any refill of the data L1 TLB from the L2 TLB. This includes refills that result in a translation fault. The following instructions are not counted: • TLB maintenance instructions This event counts regardless of whether the MMU is enabled.
0x8	[13:9]	INST_RETIRED	Instruction architecturally executed. This event counts all retired instructions, including those that fail their condition check.
0x9	[14]	EXC_TAKEN	Exception taken
0x0A	[15]	EXC_RETURN	Instruction architecturally executed, condition code check pass, exception return
0x0B	[16]	CID_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, write to CONTEXTIDR. This event only counts writes to CONTEXTIDR in AArch32 state, and via the CONTEXTIDR_EL1 mnemonic in AArch64 state. The following instructions are not counted: • Writes to CONTEXTIDR_EL12 and CONTEXTIDR_EL2
0x10	[17]	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed. This event counts any predictable branch instruction which is mispredicted either due to dynamic misprediction or because the MMU is off and the branches are statically predicted not taken.
0x11	[18]	CPU_CYCLES	Cycle
0x12	[20:19]	BR_PRED	Predictable branch speculatively executed. This event counts all predictable branches.
0x13	[23:21]	MEM_ACCESS	Data memory access. This event counts memory accesses due to load or store instructions. The following instructions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks or prefetches This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.
0x14	[24]	L1I_CACHE	L1 instruction cache access or L0 Macro-op cache access. This event counts any instruction fetch which accesses the L1 instruction cache or L0 Macro-op cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x15	[25]	L1D_CACHE_WB	L1 data cache Write-Back. This event counts any write-back of data from the L1 data cache to L2 or L3. This counts both victim line evictions and snoops, including cache maintenance operations. The following instructions are not counted: • Invalidations which do not result in data being transferred out of the L1 • Full-line writes which write to L2 without writing L1, such as write streaming mode
0x16	[29:26]	L2D_CACHE	L2 unified cache access. This event counts any transaction from L1 which looks up in the L2 cache, and any write-back from the L1 to the L2. Snoops from outside the core and cache maintenance operations are not counted.
0x17	[33:30]	L2D_CACHE_REFILL	L2 unified cache refill. This event counts any Cacheable transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes and prefetches that target this level of cache, should not be counted.
0x18	[37:34]	L2D_CACHE_WB	L2 unified cache write-back. This event counts any write-back of data from the L2 cache to outside the core. This includes snoops to the L2 which return data, regardless of whether they cause an invalidation. Invalidations from the L2 which do not write data outside of the core and snoops which return data from the L1 are not counted.
0x19	[39:38]	BUS_ACCESS	Bus access. This event 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.
0x1A	[40]	MEMORY_ERROR	Local memory error. This event counts any correctable or uncorrectable memory error (ECC or parity) in the protected core RAMs.
0x1B	[45:41]	INST_SPEC	Operation speculatively executed
0x1C	[46]	TTBR_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, write to TTBR. This event only counts writes to TTBR0/TTBR1 in AArch32 state and TTBR0_EL1/TTBR1_EL1 in AArch64 state. The following instructions are not counted: • Accesses to TTBR0_EL12/TTBR1_EL12 or TTBR0_EL2/TTBR1_EL2
0x1D	[47]	BUS_MASTER_CYCLE	Bus cycles. This event duplicates CPU_CYCLES.

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x1E	[48]	COUNTER_OVERFLOW	For odd-numbered counters, increments the count by one for each overflow of the preceding even-numbered counter. For even-numbered counters, there is no increment.
0x20	[51:49]	CACHE_ALLOCATE	L2 unified cache allocation without refill. This event counts any full cache line write into the L2 cache which does not cause a linefill, including write-backs from L1 to L2 and full-line writes which do not allocate into L1.
0x21	[55:52]	BR_RETIRED	Instruction architecturally executed, branch. This event counts all branches, taken or not. This excludes exception entries, debug entries and CCFAIL branches.
0x22	[59:56]	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch. This event counts any branch counted by BR_RETIRED which is not correctly predicted and causes a pipeline flush.
0x23	[60]	STALL_FRONTEND	No operation issued because of the frontend. The counter counts on any cycle when there are no fetched instructions available to dispatch.
0x24	[61]	STALL_BACKEND	No operation issued because of the backend. The counter counts on any cycle fetched instructions are not dispatched due to resource constraints.
0x25	[64:62]	L1D_TLB	L1 data TLB access. This event counts any load or store operation which accesses the data L1 TLB. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether the MMU is enabled.
0x26	[65]	L1I_TLB	L1 instruction TLB access. This event counts any instruction fetch which accesses the instruction L1 TLB. This event counts regardless of whether the MMU is enabled.
0x29	[66]	L3D_CACHE_ALLOCATE	Attributable L3 unified cache allocation without refill. This event counts any full cache line write into the L3 cache which does not cause a linefill, including write-backs from L2 to L3 and full-line writes which do not allocate into L2.
0x2A	[69:67]	L3D_CACHE_REFILL	Attributable L3 unified cache refill. This event counts for any cacheable read transaction returning data from the SCU for which the data source was outside the cluster. Transactions such as ReadUnique are counted here as 'read' transactions, even though they can be generated by store instructions. Prefetches and stashes that target the L3 cache are not counted.
0x2B	[70]	L3D_CACHE	Attributable L3 unified cache access. This event counts for any cacheable read transaction returning data from the SCU, or for any cacheable write to the SCU.

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x2D	[71]	L2TLB_REFILL	Attributable L2 unified TLB refill. This event counts on any refill of the L2 TLB, caused by either an instruction or data access. This event does not count if the MMU is disabled.
0x2F	[73:72]	L2TLB_REQ	Attributable L2 unified TLB access. This event counts on any access to the L2 TLB (caused by a refill of any of the L1 TLBs). This event does not count if the MMU is disabled.
0x31	[74]	REMOTE_ACCESS	Access to another socket in a multi-socket system
0x34	[75]	DTLB_WLK	Access to data TLB that caused a page table walk. This event counts on any data access which causes L2D_TLB_REFILL to count.
0x35	[76]	ITLB_WLK	Access to instruction TLB that caused a page table walk. This event counts on any instruction access which causes L2D_TLB_REFILL to count.
0x36	[79:77]	LL_CACHE_RD	Last level cache access, read. • If CPUECTLR.EXTLLC is set: This event counts any cacheable read transaction which returns a data source of 'interconnect cache'. • If CPUECTLR.EXTLLC is not set: This event is a duplicate of the L*D_CACHE_RD event corresponding to the last level of cache implemented – L3D_CACHE_RD if both per-core L2 and cluster L3 are implemented, L2D_CACHE_RD if only one is implemented, or L1D_CACHE_RD if neither is implemented.
0x37	[82:80]	LL_CACHE_MISS_RD	Last level cache miss, read. • If CPUECTLR.EXTLLC is set: This event counts any cacheable read transaction which returns a data source of 'DRAM', 'remote' or 'inter-cluster peer'. • If CPUECTLR.EXTLLC is not set: This event is a duplicate of the L*D_CACHE_REFILL_RD event corresponding to the last level of cache implemented – L3D_CACHE_REFILL_RD if both per-core L2 and cluster L3 are implemented, L2D_CACHE_REFILL_RD if only one is implemented, or L1D_CACHE_REFILL_RD if neither is implemented.
0x39	N/A	L1D_CACHE_LMISS_RD	L1 data cache long-latency miss
0x3A	N/A	OP_RETIRED	Micro-operation architecturally executed
0x3B	N/A	OP_SPEC	Micro-operation speculatively executed
0x3C	N/A	STALL	No operation sent for execution
0x3D	N/A	STALL_SLOT_BACKEND	No operation sent for execution on a slot due to the backend
0x3E	N/A	STALL_SLOT_FRONTEND	No operation sent for execution on a slot due to the frontend
0x3F	N/A	STALL_SLOT	No operation sent for execution on a slot

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x40	[84:83]	L1D_CACHE_RD	L1 data cache access, read. This event counts any load operation or page table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses
0x41	[86:85]	L1D_CACHE_WR	L1 data cache access, write. This event counts any store operation which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_WR event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses
0x42	[87]	L1D_CACHE_REFILL_RD	L1 data cache refill, read. This event counts any load operation or page table walk access which causes data to be read from outside the L1, including accesses which do not allocate into L1. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses
0x43	[88]	L1D_CACHE_REFILL_WR	L1 data cache refill, write. This event counts any store operation which causes data to be read from outside the L1, including accesses which do not allocate into L1. The following instructions are not counted: • Cache maintenance instructions and prefetches • Stores of an entire cache line, even if they make a coherency request outside the L1 • Partial cache line writes which do not allocate into the L1 cache • Non-cacheable accesses
0x44	[89]	L1D_CACHE_REFILL_INNER	L1 data cache refill, inner. This event counts any L1 data cache linefill (as counted by L1D_CACHE_REFILL) which hits in the L2 cache, L3 cache or another core in the cluster.
0x45	[90]	L1D_CACHE_REFILL_OUTER	L1 data cache refill, outer. This event counts any L1 data cache linefill (as counted by L1D_CACHE_REFILL) which does not hit in the L2 cache, L3 cache or another core in the cluster, and instead obtains data from outside the cluster.
0x46	[91]	L1D_CACHE_WB_VICTIM	L1 data cache write-back, victim
0x47	[92]	L1D_CACHE_WB_CLEAN	L1 data cache write-back cleaning and coherency
0x48	[93]	L1D_CACHE_INVAL	L1 data cache invalidate
0x4C	[94]	L1D_TLB_REFILL_RD	L1 data TLB refill, read

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x4D	[95]	L1D_TLB_REFILL_WR	L1 data TLB refill, write
0x4E	[97:96]	L1D_TLB_RD	L1 data TLB access, read
0x4F	[99:98]	L1D_TLB_WR	L1 data TLB access, write
0x50	[102:100]	CACHE_ACCESS_RD	L2 unified cache access, read. This event counts any read transaction from L1 which looks up in the L2 cache. Snoops from outside the core are not counted.
0x51	[105:103]	CACHE_ACCESS_WR	L2 unified cache access, write. This event counts any write transaction from L1 which looks up in the L2 cache or any write-back from L1 which allocates into the L2 cache. Snoops from outside the core are not counted.
0x52	[108:106]	CACHE_RD_REFILL	L2 unified cache refill, read. This event counts any cacheable read transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are counted here as 'read' transactions, even though they can be generated by store instructions.
0x53	[111:109]	CACHE_WR_REFILL	L2 unified cache refill, write. This event counts any write transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are not counted as write transactions.
0x56	[114:112]	CACHE_WRITEBACK_VICTIM	L2 unified cache write-back, victim
0x57	[117:115]	CACHE_WRITEBACK_CLEAN_COH	L2 unified cache write-back, cleaning and coherency
0x58	[120:118]	L2CACHE_INV	L2 unified cache invalidate
0x5C	[121]	L2TLB_RD_REFILL	L2 unified TLB refill, read
0x5D	[122]	L2TLB_WR_REFILL	L2 unified TLB refill, write
0x5E	[124:123]	L2TLB_RD_REQ	L2 unified TLB access, read
0x5F	[125]	L2TLB_WR_REQ	L2 unified TLB access, write
0x60	[126]	BUS_ACCESS_REQ	Bus access read. This event counts for every beat of data transferred over the read data channel between the core and the SCU.
0x61	[127]	BUS_ACCESS_RETRY	Bus access write. This event counts for every beat of data transferred over the write data channel between the core and the SCU.

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x66	[129:128]	MEM_ACCESS_RD	Data memory access, read. This event counts memory accesses due to load instructions. The following instructions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks • Prefetches
0x67	[131:130]	MEM_ACCESS_WR	Data memory access, write. This event counts memory accesses due to store instructions. The following instructions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks • Prefetches
0x68	[133:132]	UNALIGNED_LD_SPEC	Unaligned access, read
0x69	[135:134]	UNALIGNED_ST_SPEC	Unaligned access, write
0x6A	[138:136]	UNALIGNED_LDST_SPEC	Unaligned access
0x6C	[139]	LDREX_SPEC	Exclusive operation speculatively executed, LDREX or LDX
0x6D	[140]	STREX_PASS_SPEC	Exclusive operation speculatively executed, STREX or STX pass
0x6E	[141]	STREX_FAIL_SPEC	Exclusive operation speculatively executed, STREX or STX fail
0x6F	[142]	STREX_SPEC	Exclusive operation speculatively executed, STREX or STX
0x70	[147:143]	LD_SPEC	Operation speculatively executed, load
0x71	[152:148]	ST_SPEC	Operation speculatively executed, store
0x73	[157:153]	DP_SPEC	Operation speculatively executed, integer data-processing
0x74	[162:158]	ASE_SPEC	Operation speculatively executed, Advanced SIMD instruction
0x75	[167:163]	VFP_SPEC	Operation speculatively executed, floating-point instruction
0x76	[169:168]	PC_WRITE_SPEC	Operation speculatively executed, software change of the PC
0x77	[174:170]	CRYPTO_SPEC	Operation speculatively executed, Cryptographic instruction
0x78	[176:175]	BR_IMMED_SPEC	Branch speculatively executed, immediate branch
0x79	[178:177]	BR_RETURN_SPEC	Branch speculatively executed, procedure return
0x7A	[180:179]	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch
0x7C	[181]	ISB_SPEC	Barrier speculatively executed, ISB
0x7D	[183:182]	DSB_SPEC	Barrier speculatively executed, DSB
0x7E	[185:184]	DMB_SPEC	Barrier speculatively executed, DMB

Table C2-1 PMU Events (continued)

Event number	PMU event bus (to trace)	Event mnemonic	Event description
0x81	[186]	EXC_UNDEF	Counts the number of undefined exceptions taken locally
0x82	[187]	EXC_SVC	Exception taken locally, Supervisor Call
0x83	[188]	EXC_PABORT	Exception taken locally, Instruction Abort
0x84	[189]	EXC_DABORT	Exception taken locally, Data Abort and SError
0x86	[190]	EXC_IRQ	Exception taken locally, IRQ
0x87	[191]	EXC_FIQ	Exception taken locally, FIQ
0x88	[192]	EXC_SMC	Exception taken locally, Secure Monitor Call
0x8A	[193]	EXC_HVC	Exception taken locally, Hypervisor Call
0x8B	[194]	EXC_TRAP_PABORT	Exception taken, Instruction Abort not taken locally
0x8C	[195]	EXC_TRAP_DABORT	Exception taken, Data Abort or SError not taken locally
0x8D	[196]	EXC_TRAP_OTHER	Exception taken, Other traps not taken locally
0x8E	[197]	EXC_TRAP_IRQ	Exception taken, IRQ not taken locally
0x8F	[198]	EXC_TRAP_FIQ	Exception taken, FIQ not taken locally
0x90	[203:199]	RC_LD_SPEC	Release consistency operation speculatively executed, load-acquire
0x91	[208:204]	RC_ST_SPEC	Release consistency operation speculatively executed, store-release
0xA0	[209]	L3_CACHE_RD	L3 cache read
0x4004	N/A	CNT_CYCLES	Constant frequency cycles
0x4005	N/A	STALL_BACKEND_MEM	No operation sent due to the backend and memory stalls
0x4006	N/A	L1I_CACHE_LMISS	L1 instruction cache long latency miss
0x4009	N/A	L2D_CACHE_LMISS_RD	L2 unified cache long latency miss
0x400B	N/A	L3D_CACHE_LMISS_RD	L3 unified cache long latency miss

C2.4 PMU interrupts

The Cortex-A78C core asserts the **nPMUIRQ** signal when the *Performance Monitoring Unit* (PMU) generates an interrupt.

You can route this signal to an external interrupt controller for prioritization and masking. This is the only mechanism that signals this interrupt to the core.

This interrupt is also driven as a trigger input to the *Cross Trigger Interface* (CTI). For more information, see the *PMU interrupts* in the *Arm® DynamIQ™ Shared Unit MP135 Technical Reference Manual*.

C2.5 Exporting PMU events

Some of the *Performance Monitoring Unit* (PMU) events are exported to the *Embedded Trace Macrocell* (ETM) trace unit to be monitored.

——— **Note** ———

The **PMUEVENT** bus is not exported to external components. This is because the event bus cannot safely cross an asynchronous boundary when events can be generated on every cycle.

———————————

Chapter C3
Activity Monitor Unit

This chapter describes the *Activity Monitor Unit* (AMU).

It contains the following sections:

C3.1 About the AMU

The Cortex-A78C core includes activity monitoring. It has features in common with performance monitoring, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

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

The Cortex-A78C core implements seven counters in two groups, each of which is a 64-bit counter counting a fixed event. Group 0 has 4 counters 0-3, and Group 1 has 3 counters 10-12. Activity monitoring is only implemented in AArch64.

The *Activity Monitor Unit* (AMU) operation is not affected by the debug authentication signals.

C3.2 Accessing the activity monitors

The activity monitors can be accessed by:

- The system register interface for both AArch64 and AArch32 states
- Read-only memory-mapped access using the debug APB interface

C3.2.1 Access enable bit

The access enable bit for traps on accesses to activity monitor registers is required at EL2 and EL3.

In the Cortex-A78C core, the TAM[30] bit in registers ACTLR_EL2 and ACTLR_EL3 controls the activity monitor registers enable.

C3.2.2 System register access

The core implements activity monitoring in AArch64 and the activity monitors can be accessed using the `MRS` and `MSR` instructions.

C3.2.3 External memory-mapped access

Activity monitors can also be memory-mapped accessed from the *Advanced Peripheral Bus* (APB) debug interface.

In this case, the *Activity Monitor Unit* (AMU) registers just provide debug information and are read-only.

C3.3 AMU counters

The Cortex-A78C core implements four activity monitor counters, 00-03, and three auxiliary counters, 10-12. Each of the counters has the following characteristics:

- All events are counted in 64-bit wrapping counters that overflow when they wrap. There is no support for overflow status indication or interrupts.
- Any change in clock frequency, including when a *Wait For Interrupt* (WFI) and *Wait For Event* (WFE) instruction stops the clock, can affect any counter.
- Events 00-03 and auxiliary events 10-12 are fixed, and the AMEVTYPER0<n> and AMEVTYPER1<n> evtCount bits are read-only.
- The activity monitor counters are reset to zero on a Cold reset of the power domain of the core. When the core is not in reset, the *Activity Monitor Unit* (AMU) is available.

C3.4 AMU events

The following table describes the counters that are implemented in the Cortex-A78C core and the mapping to events. All events are fixed.

Table C3-1 Mapping of counters to fixed events

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

Chapter C4
Embedded Trace Macrocell

This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex-A78C core.

It contains the following sections:

C4.1 About the ETM

The *Embedded Trace Macrocell* (ETM) trace unit is a module that performs real-time instruction flow tracing based on the ETMv4 architecture. The ETM is a CoreSight component, and is an integral part of the Arm Real-time Debug solution, Arm Development Studio.

See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* for more information.

C4.2 ETM trace unit generation options and resources

The following table shows the trace generation options implemented in the Cortex-A78C ETM trace unit.

Table C4-1 ETM trace unit generation options implemented

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

The following table shows the resources implemented in the Cortex-A78C ETM trace unit.

Table C4-2 ETM trace unit resources implemented

Description	Configuration
Number of resource selection pairs implemented	8
Number of external input selectors implemented	4
Number of external inputs implemented	165, 4 CTI + 161 PMU

Table C4-2 ETM trace unit resources implemented (continued)

Description	Configuration
Number of counters implemented	2
Reduced function counter implemented	Not implemented
Number of sequencer states implemented	4
Number of Virtual Machine ID comparators implemented	1
Number of Context ID comparators implemented	1
Number of address comparator pairs implemented	4
Number of single-shot comparator controls	1
Number of core comparator inputs implemented	0
Data address comparisons implemented	Not implemented
Number of data value comparators implemented	0

C4.3 ETM trace unit functional description

This section describes the functionality of the *Embedded Trace Macrocell* (ETM) trace unit.

The following figure shows the main functional blocks of the ETM trace unit.

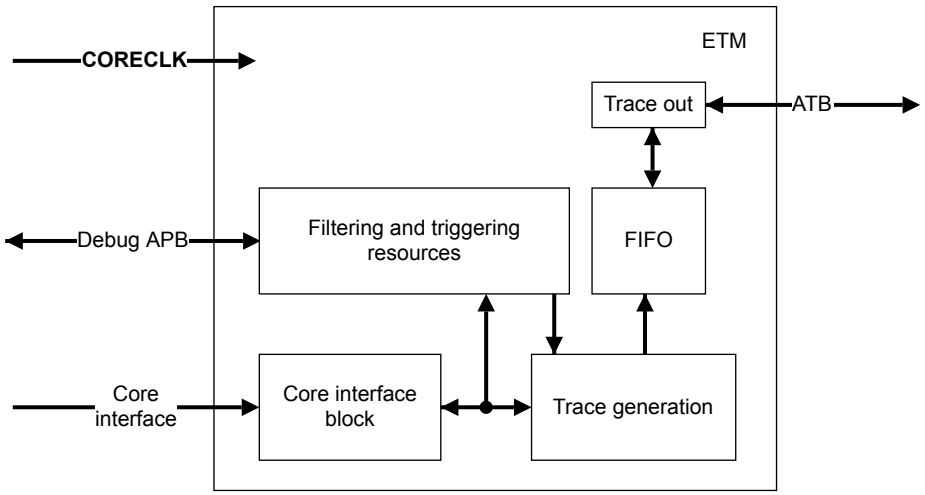

Figure C4-1 ETM functional blocks

Core interface

This block monitors the behavior of the core and generates P0 elements that are essentially executed branches and exceptions traced in program order.

Trace generation

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

Filtering and triggering resources

You can limit the amount of trace data generated by the ETM through the process of filtering.

For example, generating trace only in a certain address range. More complicated logic analyzer style filtering options are also available.

The ETM trace unit can also generate a trigger that is a signal to the Trace Capture Device to stop capturing trace.

FIFO

The trace generated by the ETM trace unit is in a highly-compressed form.

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

Trace out

Trace from FIFO is output on the *AMBA Trace Bus* (ATB) interface.

C4.4 Resetting the ETM

The reset for the *Embedded Trace Macrocell* (ETM) trace unit is the same as a Cold reset for the core.

The ETM trace unit is not reset when Warm reset is applied to the core so that tracing through Warm reset is possible. However, if the core is reset using Warm reset, the last few instructions provided by the core before the reset might not be traced.

If the ETM trace unit is reset, tracing stops until the ETM trace unit is reprogrammed and re-enabled.

C4.5 Programming and reading ETM trace unit registers

You program and read the *Embedded Trace Macrocell* (ETM) trace unit registers using the Debug APB interface.

The core does not have to be in Debug state when you program the ETM trace unit registers.

When you are programming the ETM trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition.

To disable the ETM trace unit, use the TRCPRGCTLR.EN bit.

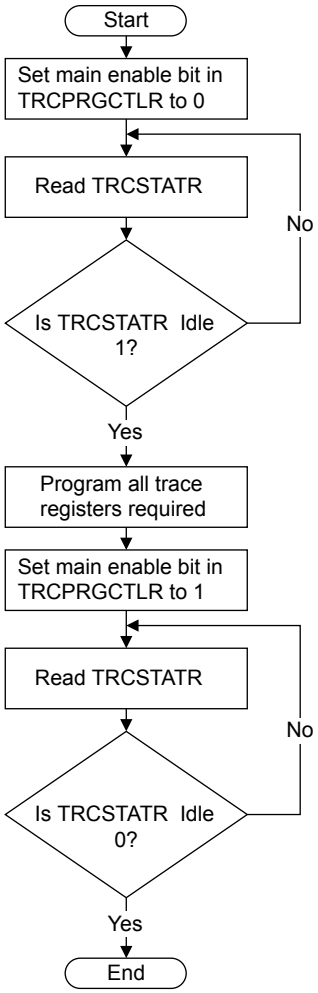

Figure C4-2 Programming ETM trace unit registers

C4.6 ETM trace unit register interfaces

The Cortex-A78C core supports only memory-mapped interface to trace registers.

See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4* for information on the behaviors on register accesses for different trace unit states and the different access mechanisms.

Related references

C1.4 External debug interface on page C1-424

C4.7 Interaction with the PMU and Debug

This section describes the interaction with the *Performance Monitoring Unit* (PMU) and the effect of debug double lock on trace register access.

Interaction with the PMU

The Cortex-A78C core includes a PMU that enables events, such as cache misses and instructions executed, to be counted over a period of time.

The PMU and *Embedded Trace Macrocell* (ETM) trace unit function together.

Use of PMU events by the ETM trace unit

The PMU architectural events described in *C2.3 PMU events* on page C2-428 are available to the ETM trace unit through the extended input facility.

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

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

Related references

C2.3 PMU events on page C2-428

Chapter C5
Statistical Profiling Extension

This chapter describes the *Statistical Profiling Extension* (SPE) for the Cortex-A78C core.

It contains the following sections:

C5.1 About the statistical profiling extension

The Cortex-A78C core supports the *Statistical Profiling Extension* (SPE), which was introduced in Armv8.2 and further enhanced in Armv8.5. SPE provides a statistical view of the performance characteristics of executed instructions, which can be used by programmers to optimize their code for better performance.

This statistical view is provided by periodically capturing profiles of the characteristics of micro-operations as they are executed on the Cortex-A78C core, and writing those profiles to memory after the corresponding instruction has retired.

——— **Note** ———

Profiling is always disabled at EL3. When profiling is disabled, no samples are collected and the sample counters are frozen.

———————————

C5.2 SPE functional description

This section describes the functionality of the *Statistical Profiling Extension* (SPE).

At a high level, SPE behavior consists of:

- Selection of the micro-operation to be profiled
- Marking the selected micro-operation throughout its lifetime in the core, indicating within the various units that it is to be profiled
- Storing data about the profiled micro-operation in internal registers during its lifetime in the core
- Following retire/abort/flush of the profiled instruction, recording the profile data to memory

While the SPE architecture allows either instructions or micro-operations to be profiled, the core will profile micro-operations in order to minimize the amount of logic necessary to support SPE.

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

The profiling activity is expected to be largely non-intrusive to the core performance, meaning the core's performance should not be meaningfully perturbed while profiling is taking place. Permitted perturbation includes using LS/L2 bandwidth to record the profile data to memory.

Note

The rate of occurrence of this activity depends on the sampling rate, which is user-specified, so it may be possible for the user to specify a sampling rate that is meaningfully intrusive to the core's performance.

- The core's recommended minimum sampling interval is once per 1024 uops.
- This value is also communicated to software via the PMSIDR_EL1 interval bits.

Unlike trace information, SPE profiles are written to memory using a *Virtual Address* (VA), which means that writes of profiles must have access to the MMU in order to translate a VA to a *Physical Address* (PA), and must have a means to be written to memory.

C5.3 IMPLEMENTATION DEFINED features of SPE

This section describes the IMPLEMENTATION DEFINED features of *Statistical Profiling Extension* (SPE).

Events definition

The Cortex-A78C core includes a 32-bit event packet which is defined in the following table.

Bit	Definition
[31:13]	Reserved
12	Late prefetch
11	Data alignment flag
10	Remote access
9	Last level cache miss
8	Last level cache access
7	Branch mispredicted
6	Not taken
5	DTLB walk
4	TLB access
3	L1 data cache refill
2	L1 data cache access
1	Architecturally retired
0	Generated exception

Data source packet

The Cortex-A78C core provides an 8-bit data source for load and store operations as defined in the following table. All other values are reserved.

Value	Name
`0b0000`	L1 data cache
`0b1000`	L2 cache
`0b1001`	Peer CPU
`0b1010`	Local cluster
`0b1011`	System cache
`0b1100`	Peer cluster
`0b1101`	Remote
`0b1110`	DRAM

Part D
Debug registers

Chapter D1
AArch32 debug registers

This chapter describes the debug registers in the AArch32 Execution state and shows examples of how to use them.

It contains the following section:

D1.1 AArch32 debug register summary

The following table summarizes the 32-bit and 64-bit debug control registers that are accessible in the AArch32 Execution state from the internal CP14 interface. These registers are accessed by the `MCR` and `MRC` instructions in the order of CRn, op2, CRm, Op1 or `MCRR` and `MRRC` instructions in the order of CRm, Op1.

For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Arm®v8, for Arm®v8-A architecture profile*.

Table D1-1 AArch32 debug register summary

CRn	Op2	CRm	Op1	Name	Type	Reset	Description
c0	0	c1	0	DBGDSCRint	RO	`000x0000`	Debug Status and Control Register, Internal View
c0	0	c5	0	DBGDTRTXint	WO	-	Debug Data Transfer Register, Transmit, Internal View
c0	0	c5	0	DBGDTRRXint	RO	`0x00000000`	Debug Data Transfer Register, Receive, Internal View

Chapter D2
AArch64 debug registers

This chapter describes the debug registers in the AArch64 Execution state and shows examples of how to use them.

It contains the following sections:

D2.1 AArch64 debug register summary

This section summarizes the debug control registers that are accessible in the AArch64 Execution state.

These registers, listed in the following table, are accessed by the `MRS` and `MSR` instructions in the order of Op0, CRn, Op1, CRm, Op2.

See *D3.1 Memory-mapped debug register summary* on page D3-474 for a complete list of registers accessible from the external debug interface. The 64-bit registers cover two addresses on the external memory interface. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table D2-1 AArch64 debug register summary

Name	Type	Reset	Width	Description
OSDTRRX_EL1	RW	`0x00000000`	32	Debug Data Transfer Register, Receive, External View
DBGBVR0_EL1	RW	-	64	Debug Breakpoint Value Register 0
DBGBCR0_EL1	RW	UNK	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
DBGWVR0_EL1	RW	-	64	Debug Watchpoint Value Register 0
DBGWCR0_EL1	RW	UNK	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470
DBGBVR1_EL1	RW	-	64	Debug Breakpoint Value Register 1
DBGBCR1_EL1	RW	UNK	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
DBGWVR1_EL1	RW	-	64	Debug Watchpoint Value Register 1
DBGWCR1_EL1	RW	UNK	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470
MDCCINT_EL1	RW	`0x00000000`	32	Monitor Debug Comms Channel Interrupt Enable Register
MDSCR_EL1	RW	-	32	Monitor Debug System Control Register, EL1
DBGBVR2_EL1	RW	-	64	Debug Breakpoint Value Register 2
DBGBCR2_EL1	RW	UNK	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
DBGWVR2_EL1	RW	-	64	Debug Watchpoint Value Register 2
DBGWCR2_EL1	RW	UNK	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470
OSDTRTX_EL1	RW	-	32	Debug Data Transfer Register, Transmit, External View
DBGBVR3_EL1	RW	-	64	Debug Breakpoint Value Register 3
DBGBCR3_EL1	RW	UNK	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
DBGWVR3_EL1	RW	-	64	Debug Watchpoint Value Register 3
DBGWCR3_EL1	RW	UNK	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470
DBGBVR4_EL1	RW	-	64	Debug Breakpoint Value Register 4

Table D2-1 AArch64 debug register summary (continued)

Name	Type	Reset	Width	Description
DBGBCR4_EL1	RW	UNK	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
DBGBVR5_EL1	RW	-	64	Debug Breakpoint Value Register 5
DBGBCR5_EL1	RW	UNK	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
OSECCR_EL1	RW	0x00000000	32	Debug OS Lock Exception Catch Register
MDCCSR_EL0	RO	0x00000000	32	Monitor Debug Comms Channel Status Register
DBGDTR_EL0	RW	0x00000000	64	Debug Data Transfer Register, half-duplex
DBGDTRTX_EL0	WO	0x00000000	32	Debug Data Transfer Register, Transmit, Internal View
DBGDTRRX_EL0	RO	0x00000000	32	Debug Data Transfer Register, Receive, Internal View
MDRAR_EL1	RO	-	64	Debug ROM Address Register. This register is reserved, RES0.
OSLAR_EL1	WO	-	32	Debug OS Lock Access Register
OSLSR_EL1	RO	0x0000000A	32	Debug OS Lock Status Register
OSDLR_EL1	RW	0x00000000	32	Debug OS Double Lock Register
DBGPRCR_EL1	RW	-	32	Debug Power/Reset Control Register
DBGCLAIMSET_EL1	RW	0x000000FF	32	*D2.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1* on page D2-469
DBGCLAIMCLR_EL1	RW	0x00000000	32	Debug Claim Tag Clear Register
DBGAUTHSTATUS_EL1	RO	0x000000AA	32	Debug Authentication Status Register

D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1

The DBGBCRn_EL1 holds control information for a breakpoint. Each DBGBVR_EL1 is associated with a DBGBCR_EL1 to form a *Breakpoint Register Pair* (BRP). DBGBVRn_EL1 is associated with DBGBCRn_EL1 to form BRPn. The range of *n* for DBGBCRn_EL1 is 0 to 5.

Bit field descriptions

The DBGBCRn_EL1 registers are 32-bit registers.

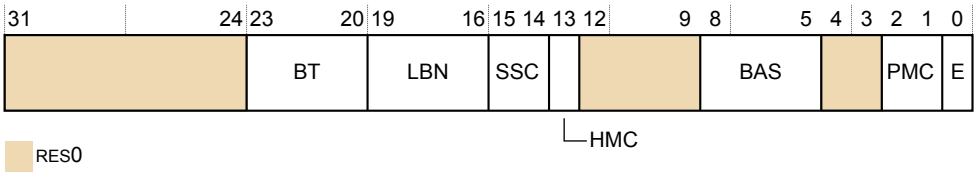

Figure D2-1 DBGBCRn_EL1 bit assignments

RES0, [31:24]

RES0 Reserved

BT, [23:20]

Breakpoint Type. This field controls the behavior of Breakpoint debug event generation. This includes the meaning of the value held in the associated DBGBVR*n*_EL1, indicating whether it is an instruction address match or mismatch, or a Context match. It also controls whether the breakpoint is linked to another breakpoint. The possible values are:

`0b0000` Unlinked instruction address match
`0b0001` Linked instruction address match
`0b0010` Unlinked Context ID match
`0b0011` Linked Context ID match
`0b0100` Unlinked instruction address mismatch
`0b0101` Linked instruction address mismatch
`0b0110` Unlinked CONTEXTIDR_EL1 match
`0b0111` Linked CONTEXTIDR_EL1 match
`0b1000` Unlinked VMID match
`0b1001` Linked VMID match
`0b1010` Unlinked VMID + Conext ID match
`0b1011` Linked VMID + Context ID match
`0b1100` Unlinked CONTEXTIDR_EL2 match
`0b1101` Linked CONTEXTIDR_EL2 match
`0b1110` Unlinked Full Context ID match
`0b1111` Linked Full Context ID match.

The field break down is:

- BT[3:1]: Base type. If the breakpoint is not context-aware, these bits are RES0. Otherwise, the possible values are:

  `0b000` Match address. DBGBVR*n*_EL1 is the address of an instruction.
  `0b001` Match context ID. DBGBVR*n*_EL1[31:0] is a context ID.
  `0b010` Match VMID. DBGBVR*n*_EL1[47:32] is a VMID.

`0b011` Match VMID and CONTEXTIDR_EL1. DBGBVR*n*_EL1[31:0] is a context ID, and DBGBVR*n*_EL1[47:32] is a VMID.

- BT[2]: Mismatch. RES0.
- BT[0]: Enable linking.

LBN, [19:16]

Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

SSC, [15:14]

Security State Control. Determines the Security states under which a Breakpoint debug event for breakpoint *n* is generated.

This field must be interpreted with the *Higher Mode Control* (HMC), and *Privileged Mode Control* (PMC), fields to determine the mode and security states that can be tested.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for possible values of the HMC and PMC fields.

HMC, [13]

Hyp Mode Control bit. Determines the debug perspective for deciding when a breakpoint debug event for breakpoint *n* is generated.

This bit must be interpreted with the SSC and PMC fields to determine the mode and security states that can be tested.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for possible values of the SSC and PMC fields.

RES0, [12:9]

RES0 Reserved

BAS, [8:5]

Byte Address Select. Defines which half-words a regular breakpoint matches, regardless of the instruction set and Execution state. A debugger must program this field as follows:

`0x3` Match the T32 instruction at DBGBVR*n*_EL1

`0xC` Match the T32 instruction at DBGBVR*n*_EL1+2

`0xF` Match the A64 or A32 instruction at DBGBVR*n*_EL1, or context match

All other values are reserved.

The Armv8-A architecture does not support direct execution of Java bytecodes. BAS[3] and BAS[1] ignore writes and on reads return the values of BAS[2] and BAS[0] respectively.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on how the BAS field is interpreted by hardware.

RES0, [4:3]

RES0 Reserved

PMC, [2:1]

Privileged Mode Control. Determines the Exception level or levels that a breakpoint debug event for breakpoint *n* is generated.

This field must be interpreted with the SSC and HMC fields to determine the mode and security states that can be tested.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for possible values of the SSC and HMC fields.

Bits[2:1] have no effect for accesses made in Hyp mode.

E, [0]

Enable breakpoint. This bit enables the BRP:

`0` BRP disabled

`1` BRP enabled

A BRP never generates a breakpoint debug event when it is disabled.

The value of DBGBCR*n*_EL1.E is UNKNOWN on reset. A debugger must ensure that DBGBCR*n*_EL1.E has a defined value before it enables debug.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

D2.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1

The DBGCLAIMSET_EL1 is used by software to set CLAIM bits to 1.

Bit field descriptions

The DBGCLAIMSET_EL1 is a 32-bit register.

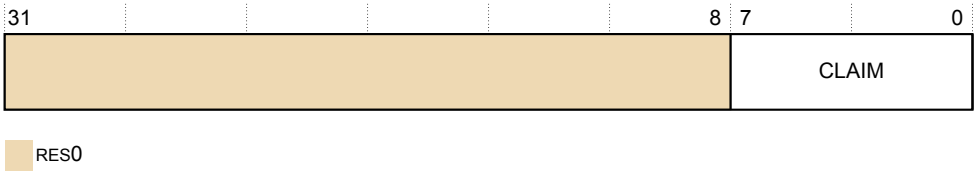

Figure D2-2 DBGCLAIMSET_EL1 bit assignments

RES0, [31:8]

RES0 Reserved

CLAIM, [7:0]

Claim set bits

Writing a 1 to one of these bits sets the corresponding CLAIM bit to 1. This is an indirect write to the CLAIM bits.

A single write operation can set multiple bits to 1. Writing 0 to one of these bits has no effect.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1

The DBGWCRn_EL1 holds control information for a watchpoint. Each DBGWCR_EL1 is associated with a DBGWVR_EL1 to form a *Watchpoint Register Pair* (WRP). DBGWCRn_EL1 is associated with DBGWVRn_EL1 to form WRP*n*. The range of *n* for DBGBCRn_EL1 is 0 to 3.

Bit field descriptions

The DBGWCRn_EL1 registers are 32-bit registers.

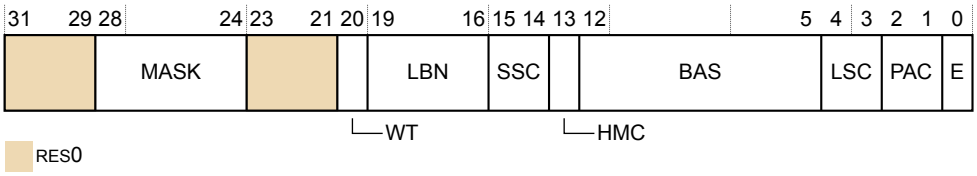

Figure D2-3 DBGWCRn_EL1 bit assignments

RES0, [31:29]

RES0 Reserved

MASK, [28:24]

Address mask. Only objects up to 2GB can be watched using a single mask.

`0b00000` No mask

`0b00001` Reserved

`0b00010` Reserved

Other values mask the corresponding number of address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).

RES0, [23:21]

RES0 Reserved

WT, [20]

Watchpoint type. Possible values are:

`0b0` Unlinked data address match

`0b1` Linked data address match

On Cold reset, the field reset value is architecturally UNKNOWN.

LBN, [19:16]

Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

On Cold reset, the field reset value is architecturally UNKNOWN.

SSC, [15:14]

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.

On Cold reset, the field reset value is architecturally UNKNOWN.

HMC, [13]

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.

On Cold reset, the field reset value is architecturally UNKNOWN.

BAS, [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR*n*_EL1 is being watched. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

LSC, [4:3]

Load/store access control. This field enables watchpoint matching on the type of access being made. The possible values are:

`0b01` Match instructions that load from a watchpoint address

`0b10` Match instructions that store to a watchpoint address

`0b11` Match instructions that load from or store to a watchpoint address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

IGNORED if E is 0.

On Cold reset, the field reset value is architecturally UNKNOWN.

PAC, [2:1]

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 HMCfields.

On Cold reset, the field reset value is architecturally UNKNOWN.

E, [0]

Enable watchpoint n. Possible values are:

`0b0` Watchpoint disabled

`0b1` Watchpoint enabled

On Cold reset, the field reset value is architecturally UNKNOWN.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Chapter D3
Memory-mapped debug registers

This chapter describes the memory-mapped debug registers and shows examples of how to use them.

It contains the following sections:

D3.1 Memory-mapped debug register summary

The following table shows the offset address for the registers that are accessible from the external debug interface.

For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table D3-1 Memory-mapped debug register summary

Offset	Name	Type	Width	Description
0x000-0x01C	-	-	-	Reserved
0x020	EDESR	RW	32	External Debug Event Status Register
0x024	EDECR	RW	32	External Debug Execution Control Register
0x028-0x02C	-	-	-	Reserved
0x030	EDWAR[31:0]	RO	64	External Debug Watchpoint Address Register
0x034	EDWAR[63:32]			
0x038-0x07C	-	-	-	Reserved
0x080	DBGDTRRX_EL0	RW	32	Debug Data Transfer Register, Receive
0x084	EDITR	WO	32	External Debug Instruction Transfer Register
0x088	EDSCR	RW	32	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	WO	32	Debug Data Transfer Register, Transmit
0x090	EDRCR	WO	32	*D3.14 EDRCR, External Debug Reserve Control Register* on page D3-490
0x094	-	RW	32	Reserved
0x098	EDECCR	RW	32	External Debug Exception Catch Control Register
0x09C	-	-	-	Reserved
0x0A0	-	-	-	Reserved
0x0A4	-	-	-	Reserved
0x0A8	-	-	-	Reserved
0x0AC	-	-	-	Reserved
0x0B0-0x2FC	-	-	-	Reserved
0x300	OSLAR_EL1	WO	32	OS Lock Access Register
0x304-0x30C	-	-	-	Reserved
0x310	EDPRCR	RW	32	External Debug Power/Reset Control Register
0x314	EDPRSR	RO	32	External Debug Processor Status Register
0x318-0x3FC	-	-	-	Reserved
0x400	DBGBVR0_EL1[31:0]	RW	64	Debug Breakpoint Value Register 0
0x404	DBGBVR0_EL1[63:32]			

Table D3-1 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0x408	DBGBCR0_EL1	RW	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
0x40C	-	-	-	Reserved
0x410	DBGBVR1_EL1[31:0]	RW	64	Debug Breakpoint Value Register 1
0x414	DBGBVR1_EL1[63:32]			
0x418	DBGBCR1_EL1	RW	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
0x41C	-	-	-	Reserved
0x420	DBGBVR2_EL1[31:0]	RW	64	Debug Breakpoint Value Register 2
0x424	DBGBVR2_EL1[63:32]			
0x428	DBGBCR2_EL1	RW	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
0x42C	-	-	-	Reserved
0x430	DBGBVR3_EL1[31:0]	RW	64	Debug Breakpoint Value Register 3
0x434	DBGBVR3_EL1[63:32]			
0x438	DBGBCR3_EL1	RW	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
0x43C	-	-	-	Reserved
0x440	DBGBVR4_EL1[31:0]	RW	64	Debug Breakpoint Value Register 4
0x444	DBGBVR4_EL1[63:32]			
0x448	DBGBCR4_EL1	RW	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
0x44C	-	-	-	Reserved
0x450	DBGBVR5_EL1[31:0]	RW	64	Debug Breakpoint Value Register 5
0x454	DBGBVR5_EL1[63:32]			
0x458	DBGBCR5_EL1	RW	32	*D2.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1* on page D2-466
0x45C-0x7FC	-	-	-	Reserved
0x800	DBGWVR0_EL1[31:0]	RW	64	Debug Watchpoint Value Register 0
0x804	DBGWVR0_EL1[63:32]			
0x808	DBGWCR0_EL1	RW	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470
0x80C	-	-	-	Reserved
0x810	DBGWVR1_EL1[31:0]	RW	64	Debug Watchpoint Value Register 1
0x814	DBGWVR1_EL1[63:32]			
0x818	DBGWCR1_EL1	RW	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470

Table D3-1 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0x81C	-	-	-	Reserved
0x820	DBGWVR2_EL1[31:0]	RW	64	Debug Watchpoint Value Register 2
0x824	DBGWVR2_EL1[63:32]			
0x828	DBGWCR2_EL1	RW	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470
0x82C	-	-	-	Reserved
0x830	DBGWVR3_EL1[31:0]	RW	64	Debug Watchpoint Value Register 0,
0x834	DBGWVR3_EL1[63:32]			
0x838	DBGWCR3_EL1	RW	32	*D2.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1* on page D2-470
0x83C-0xCFC	-	-	-	Reserved
0xD00	MIDR	RO	32	*B2.107 MIDR_EL1, Main ID Register, EL1* on page B2-313
0xD04-0xD1C	-	-	-	Reserved
0xD20	EDPFR[31:0]	RO	64	*B2.84 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1* on page B2-273
0xD24	EDPFR[63:32]			
0xD28	EDDFR[31:0]	RO	64	*B2.84 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1* on page B2-273
0xD2C	EDDFR[63:32]			
0xD60-0xEFC	-	-	-	Reserved
0xF00	-	-	-	Reserved
0xF04-0xF9C	-	-	-	Reserved
0xFA0	DBGCLAIMSET_EL1	RW	32	*D2.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1* on page D2-469
0xFA4	DBGCLAIMCLR_EL1	RW	32	Debug Claim Tag Clear Register
0xFA8	EDDEVAFF0	RO	32	External Debug Device Affinity Register 0
0xFAC	EDDEVAFF1	RO	32	External Debug Device Affinity Register 1
0xFB0	-	-	-	Reserved
0xFB4	-	-	-	Reserved
0xFB8	DBGAUTHSTATUS_EL1	RO	32	Debug Authentication Status Register
0xFBC	EDDEVARCH	RO	32	External Debug Device Architecture Register
0xFC0	EDDEVID2	RO	32	External Debug Device ID Register 2, RES0
0xFC4	EDDEVID1	RO	32	*D3.7 EDDEVID1, External Debug Device ID Register 1* on page D3-483
0xFC8	EDDEVID	RO	32	*D3.6 EDDEVID, External Debug Device ID Register 0* on page D3-482
0xFCC	EDDEVTYPE	RO	32	External Debug Device Type Register
0xFD0	EDPIDR4	RO	32	*D3.12 EDPIDR4, External Debug Peripheral Identification Register 4* on page D3-488

Table D3-1 Memory-mapped debug register summary (continued)

Offset	Name	Type	Width	Description
0xFD4-0xFDC	EDPIDR5-7	RO	32	*D3.13 EDPIDRn, External Debug Peripheral Identification Registers 5-7* on page D3-489
0xFE0	EDPIDR0	RO	32	*D3.8 EDPIDR0, External Debug Peripheral Identification Register 0* on page D3-484
0xFE4	EDPIDR1	RO	32	*D3.9 EDPIDR1, External Debug Peripheral Identification Register 1* on page D3-485
0xFE8	EDPIDR2	RO	32	*D3.10 EDPIDR2, External Debug Peripheral Identification Register 2* on page D3-486
0xFEC	EDPIDR3	RO	32	*D3.11 EDPIDR3, External Debug Peripheral Identification Register 3* on page D3-487
0xFF0	EDCIDR0	RO	32	*D3.2 EDCIDR0, External Debug Component Identification Register 0* on page D3-478
0xFF4	EDCIDR1	RO	32	*D3.3 EDCIDR1, External Debug Component Identification Register 1* on page D3-479
0xFF8	EDCIDR2	RO	32	*D3.4 EDCIDR2, External Debug Component Identification Register 2* on page D3-480
0xFFC	EDCIDR3	RO	32	*D3.5 EDCIDR3, External Debug Component Identification Register 3* on page D3-481

D3.2 EDCIDR0, External Debug Component Identification Register 0

The EDCIDR0 provides information to identify an external debug component.

Bit field descriptions

The EDCIDR0 is a 32-bit register.

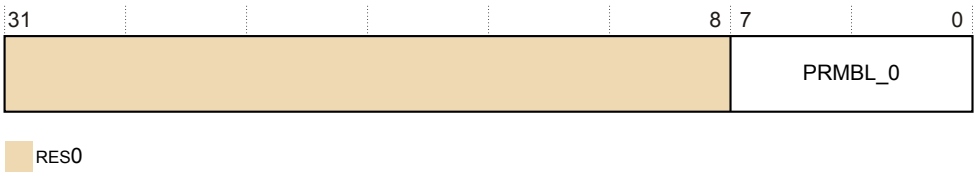

Figure D3-1 EDCIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_0, [7:0]

`0x0D` Preamble byte 0

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

D3.3 EDCIDR1, External Debug Component Identification Register 1

The EDCIDR1 provides information to identify an external debug component.

Bit field descriptions

The EDCIDR1 is a 32-bit register.

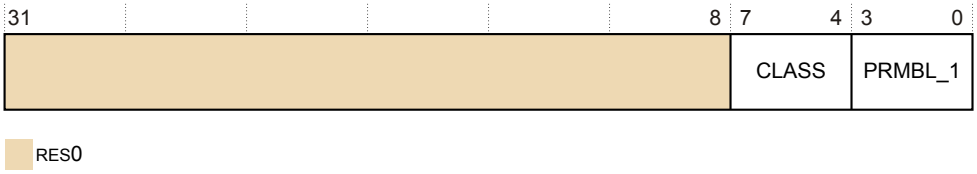

Figure D3-2 EDCIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

CLASS, [7:4]

`0x9` Debug component

PRMBL_1, [3:0]

`0x0` Preamble

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

D3.4 EDCIDR2, External Debug Component Identification Register 2

The EDCIDR2 provides information to identify an external debug component.

Bit field descriptions

The EDCIDR2 is a 32-bit register.

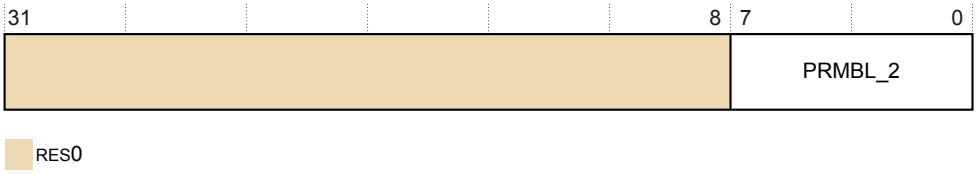

Figure D3-3 EDCIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_2, [7:0]

`0x05` Preamble byte 2

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

D3.5 EDCIDR3, External Debug Component Identification Register 3

The EDCIDR3 provides information to identify an external debug component.

Bit field descriptions

The EDCIDR3 is a 32-bit register.

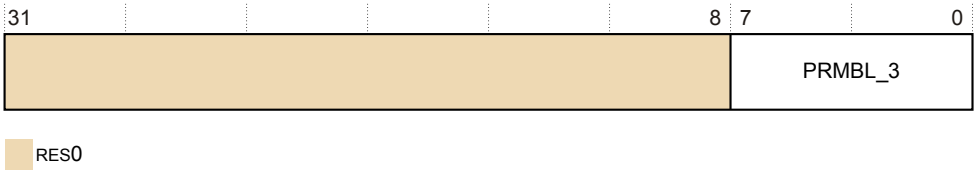

Figure D3-4 EDCIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_3, [7:0]

`0xB1` Preamble byte 3

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

D3.6 EDDEVID, External Debug Device ID Register 0

The EDDEVID provides extra information for external debuggers about features of the debug implementation.

Bit field descriptions

The EDDEVID is a 32-bit register.

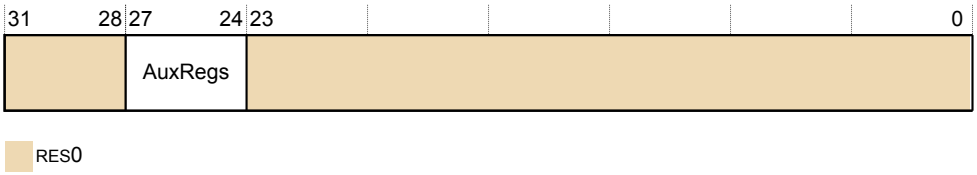

Figure D3-5 EDDEVID bit assignments

RES0, [31:28]

RES0 Reserved

AuxRegs, [27:24]

Indicates support for Auxiliary registers:

`0x0` None supported

RES0, [23:0]

RES0 Reserved

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDDEVID can be accessed through the external debug interface, offset `0xFC8`.

D3.7 EDDEVID1, External Debug Device ID Register 1

The EDDEVID1 provides extra information for external debuggers about features of the debug implementation.

Bit field descriptions

The EDDEVID1 is a 32-bit register.

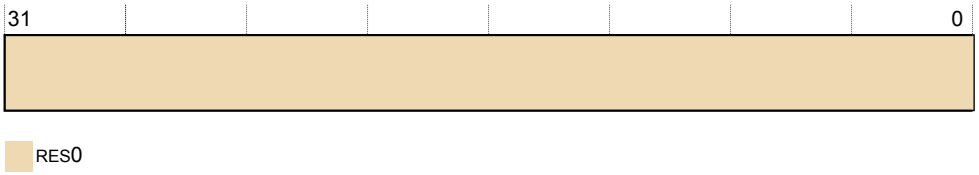

Figure D3-6 EDDEVID1 bit assignments

RES0, [31:0]

RES0 Reserved

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDDEVID1 can be accessed through the external debug interface, offset `0xFC4`.

D3.8 EDPIDR0, External Debug Peripheral Identification Register 0

The EDPIDR0 provides information to identify an external debug component.

Bit field descriptions

The EDPIDR0 is a 32-bit register.

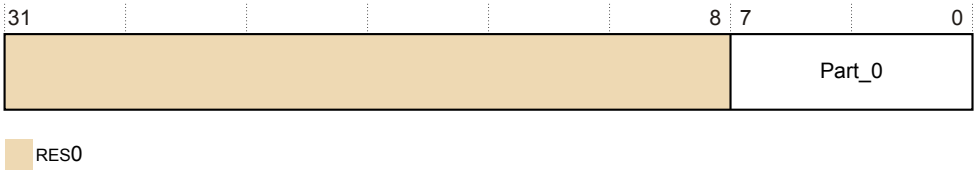

Figure D3-7 EDPIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

Part_0, [7:0]

`0x4B` Least significant byte of the debug part number

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

D3.9 EDPIDR1, External Debug Peripheral Identification Register 1

The EDPIDR1 provides information to identify an external debug component.

Bit field descriptions

The EDPIDR1 is a 32-bit register.

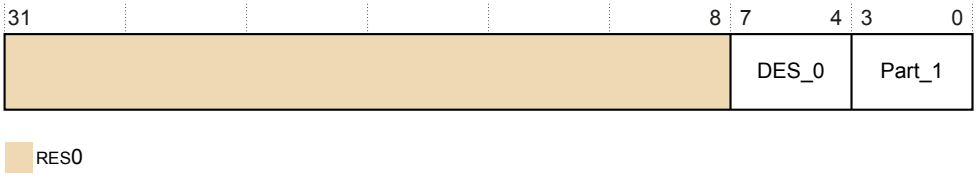

Figure D3-8 EDPIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

DES_0, [7:4]

`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.

Part_1, [3:0]

`0xD` Most significant nibble of the debug part number

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

D3.10 EDPIDR2, External Debug Peripheral Identification Register 2

The EDPIDR2 provides information to identify an external debug component.

Bit field descriptions

The EDPIDR2 is a 32-bit register.

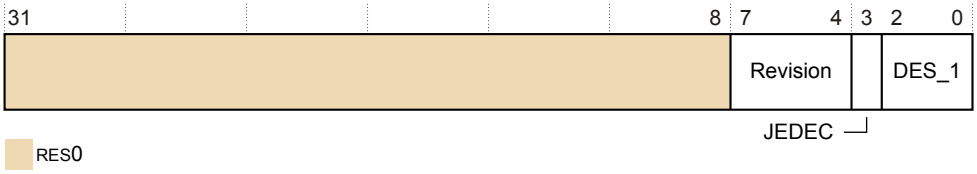

Figure D3-9 EDPIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

Revision, [7:4]

`0` r0p1

JEDEC, [3]

`0b1` RAO. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is the most significant nibble of JEP106 ID code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

D3.11 EDPIDR3, External Debug Peripheral Identification Register 3

The EDPIDR3 provides information to identify an external debug component.

Bit field descriptions

The EDPIDR3 is a 32-bit register.

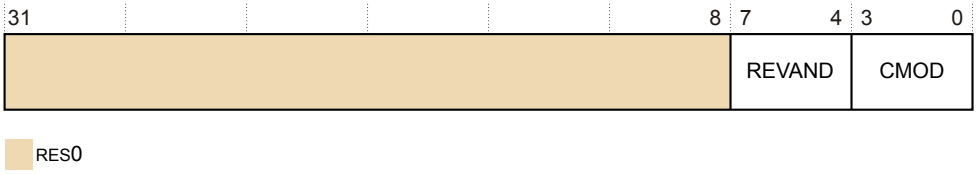

Figure D3-10 EDPIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

REVAND, [7:4]

`0x0` Part minor revision

CMOD, [3:0]

`0x0` Customer modified

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

D3.12 EDPIDR4, External Debug Peripheral Identification Register 4

The EDPIDR4 provides information to identify an external debug component.

Bit field descriptions

The EDPIDR4 is a 32-bit register.

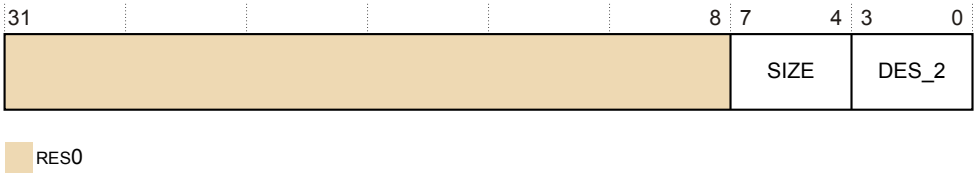

Figure D3-11 EDPIDR4 bit assignments

RES0, [31:8]

RES0 Reserved

SIZE, [7:4]

`0x0` Size of the component. Log_2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited This is the least significant nibble JEP106 continuation code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

D3.13 EDPIDRn, External Debug Peripheral Identification Registers 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

D3.14 EDRCR, External Debug Reserve Control Register

The EDRCR is part of the Debug registers functional group. This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

Bit field descriptions

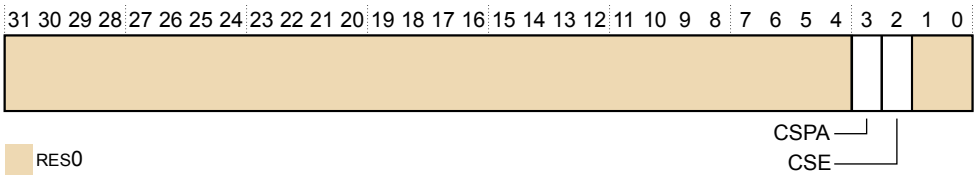

Figure D3-12 EDRCR bit assignments

RES0, [31:4]

RES0 Reserved

CSPA, [3]

Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv bit to 0. The actions on writing to this bit are:

`0` No action

`1` Clear the EDSCR.PipeAdv bit to 0

CSE, [2]

Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0. The actions on writing to this bit are:

`0` No action

`1` Clear the EDSCR.{TXU, RXO, ERR} bits, and, if the core is in Debug state, the EDSCR.ITO bit, to 0

RES0, [1:0]

RES0 Reserved

The EDRCR can be accessed through the internal memory-mapped interface and the external debug interface, offset `0x090`.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	WI	WO

Configurations

EDRCR is in the Core power domain.

Chapter D4
AArch32 PMU registers

This chapter describes the AArch32 *Performance Monitoring Unit* (PMU) registers and shows examples of how to use them.

It contains the following sections:

D4.1 AArch32 PMU register summary

The PMU counters and their associated control registers are accessible in the AArch32 Execution state from the internal CP15 system register interface with `MCR` and `MRC` instructions for 32-bit registers and `MCRR` and `MRRC` for 64-bit registers.

The following table gives a summary of the Cortex-A78C PMU registers in the AArch32 Execution state. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table D4-1 PMU register summary in the AArch32 Execution state

CRn	Op1	CRm	Op2	Name	Type	Width	Reset	Description
c9	0	c12	0	PMCR	RW	32	`0x412130XX`	*D4.5 PMCR, Performance Monitors Control Register* on page D4-502
c9	0	c12	1	PMCNTENSET	RW	32	`0x00000000`	Performance Monitors Count Enable Set Register
c9	0	c12	2	PMCNTENCLR	RW	32	`0x00000000`	Performance Monitors Count Enable Clear Register
c9	0	c12	3	PMOVSR	RW	32	`0x00000000`	Performance Monitors Overflow Flag Status Register
c9	0	c12	4	PMSWINC	WO	32	UNK	Performance Monitors Software Increment Register
c9	0	c12	5	PMSELR	RW	32	UNK	Performance Monitors Event Counter Selection Register
c9	0	c12	6	PMCEID0	RO	32	`0x7FFF0F3F`	*D4.2 PMCEID0, Performance Monitors Common Event Identification Register 0* on page D4-494
c9	0	c12	7	PMCEID1	RO	32	`0xFEF2AE7F`	*D4.3 PMCEID1, Performance Monitors Common Event Identification Register 1* on page D4-497
c9	0	c14	4	PMCEID2	RO	32	`0x00000A7F`	*D4.4 PMCEID2, Performance Monitors Common Event Identification Register 2* on page D4-500
c9	0	c14	5	PMCEID3	RO	32	`0x00000000`	Reserved
c9	0	c13	0	PMCCNTR[31:0]	RW	32	UNK	Performance Monitors Cycle Count Register
c9	3	c13	0	PMCCNTR[63:0]	RW	64	UNK	
c9	0	c13	1	PMXEVTYPER	RW	32	UNK	Performance Monitors Selected Event Type Register
c9	0	c13	2	PMXEVCNTR	RW	32	UNK	Performance Monitors Selected Event Count Register
c9	0	c14	0	PMUSERENR	RW	32	UNK	Performance Monitors User Enable Register
c9	0	c14	3	PMOVSSET	RW	32	`0x00000000`	Performance Monitor Overflow Flag Status Set Register
c14	0	c8	0	PMEVCNTR0	RW	32	UNK	Performance Monitor Event Count Registers
c14	0	c8	1	PMEVCNTR1	RW	32	UNK	
c14	0	c8	2	PMEVCNTR2	RW	32	UNK	
c14	0	c8	3	PMEVCNTR3	RW	32	UNK	
c14	0	c8	4	PMEVCNTR4	RW	32	UNK	
c14	0	c8	5	PMEVCNTR5	RW	32	UNK	

Table D4-1 PMU register summary in the AArch32 Execution state (continued)

CRn	Op1	CRm	Op2	Name	Type	Width	Reset	Description
c14	0	c12	0	PMEVTYPER0	RW	32	UNK	Performance Monitors Event Type Registers
c14	0	c12	1	PMEVTYPER1	RW	32	UNK	
c14	0	c12	2	PMEVTYPER2	RW	32	UNK	
c14	0	c12	3	PMEVTYPER3	RW	32	UNK	
c14	0	c12	4	PMEVTYPER4	RW	32	UNK	
c14	0	c12	5	PMEVTYPER5	RW	32	UNK	
c14	0	c15	7	PMCCFILTR	RW	32	UNK	Performance Monitors Cycle Count Filter Register

D4.2 PMCEID0, Performance Monitors Common Event Identification Register 0

The PMCEID0 defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

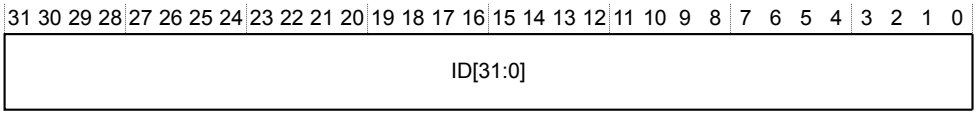

Figure D4-1 PMCEID0 bit assignments

ID[31:0], [31:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

The following table shows the PMCEID0 bit assignments with event implemented or not implemented when the associated bit is set to 1 or 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about these events.

Table D4-2 PMU events

Bit	Event mnemonic	Description
[31]	L1D_CACHE_ALLOCATE	L1 Data cache allocate: `0` This event is not implemented.
[30]	CHAIN	Chain. For odd-numbered counters, counts once for each overflow of the preceding even-numbered counter. For even-numbered counters, does not count: `1` This event is implemented.
[29]	BUS_CYCLES	Bus cycle: `1` This event is implemented.
[28]	TTBR_WRITE_RETIRED	TTBR write, architecturally executed, condition check pass - write to translation table base: `1` This event is implemented.
[27]	INST_SPEC	Instruction speculatively executed: `1` This event is implemented.
[26]	MEMORY_ERROR	Local memory error: `1` This event is implemented.
[25]	BUS_ACCESS	Bus access: `1` This event is implemented.
[24]	L2D_CACHE_WB	L2 unified cache Write-Back: `1` This event is implemented.

Table D4-2 PMU events (continued)

Bit	Event mnemonic	Description
[23]	L2D_CACHE_REFILL	L2 unified cache refill: 1 This event is implemented.
[22]	L2D_CACHE	L2 unified cache access: 1 This event is implemented.
[21]	L1D_CACHE_WB	L1 Data cache Write-Back: 1 This event is implemented.
[20]	L1I_CACHE	L1 Instruction cache access: 1 This event is implemented.
[19]	MEM_ACCESS	Data memory access: 1 This event is implemented.
[18]	BR_PRED	Predictable branch speculatively executed: 1 This event is implemented.
[17]	CPU_CYCLES	Cycle: 1 This event is implemented.
[16]	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed: 1 This event is implemented.
[15]	UNALIGNED_LDST_RETIRED	Instruction architecturally executed, condition check pass - unaligned load or store: 0 This event is not implemented.
[14]	BR_RETURN_RETIRED	Instruction architecturally executed, condition check pass - procedure return: 0 This event is not implemented.
[13]	BR_IMMED_RETIRED	Instruction architecturally executed - immediate branch: 0 This event is not implemented.
[12]	PC_WRITE_RETIRED	Instruction architecturally executed, condition check pass - software change of the PC: 0 This event is not implemented.
[11]	CID_WRITE_RETIRED	Instruction architecturally executed, condition check pass - write to CONTEXTIDR: 1 This event is implemented.
[10]	EXC_RETURN	Instruction architecturally executed, condition check pass - exception return: 1 This event is implemented.

Table D4-2 PMU events (continued)

Bit	Event mnemonic	Description
[9]	EXC_TAKEN	Exception taken: `1` This event is implemented.
[8]	INST_RETIRED	Instruction architecturally executed: `1` This event is implemented.
[7]	ST_RETIRED	Instruction architecturally executed, condition check pass - store: `0` This event is not implemented.
[6]	LD_RETIRED	Instruction architecturally executed, condition check pass - load: `0` This event is not implemented.
[5]	L1D_TLB_REFILL	L1 Data TLB refill: `1` This event is implemented.
[4]	L1D_CACHE	L1 Data cache access: `1` This event is implemented.
[3]	L1D_CACHE_REFILL	L1 Data cache refill: `1` This event is implemented.
[2]	L1I_TLB_REFILL	L1 Instruction TLB refill: `1` This event is implemented.
[1]	L1I_CACHE_REFILL	L1 Instruction cache refill: `1` This event is implemented.
[0]	SW_INCR	Instruction architecturally executed, condition check pass - software increment: `1` This event is implemented.

Note

The PMU events implemented in the above table can be found in *Table C2-1 PMU Events on page C2-428*.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

D4.3 PMCEID1, Performance Monitors Common Event Identification Register 1

The PMCEID1 defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

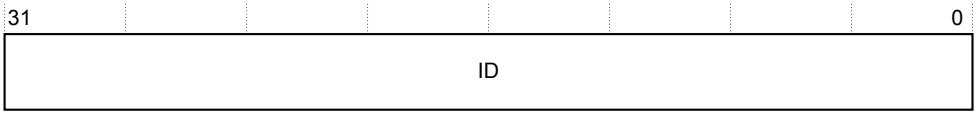

Figure D4-2 PMCEID1 bit assignments

ID[63:32], [31:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

For each bit described in the following table, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

Table D4-3 PMU common events

Bit	Event mnemonic	Description
[31]	STALL_SLOT	No operation sent for execution on a slot. `1` This event is implemented.
[30]	STALL_SLOT_FRONTEND	No operation sent for execution on a slot due to the frontend. `1` This event is implemented.
[29]	STALL_SLOT_BACKEND	No operation sent for execution on a slot due to the backend. `1` This event is implemented.
[28]	STALL	No operation sent for execution. `1` This event is implemented.
[27]	OP_SPEC	Micro-operation speculatively executed. `1` This event is implemented.
[26]	OP_RETIRED	Micro-operation architecturally executed. `1` This event is implemented.
[25]	L1D_CACHE_LMISS_RD	L1 data cache long-latency read miss. `1` This event is implemented.
[24]	REMOTE_ACCESS_RD	Attributable memory read access to another socket in a multi-socket system. `0` This event is not implemented.
[23]	LL_CACHE_MISS_RD	Attributable last level cache memory read miss. `1` This event is implemented.

Table D4-3 PMU common events (continued)

Bit	Event mnemonic	Description
[22]	LL_CACHE_RD	Attributable last level cache memory read. `1` This event is implemented.
[21]	ITLB_WALK	Attributable instruction TLB access with at least one translation table walk. `1` This event is implemented.
[20]	DTLB_WALK	Attributable data or unified TLB access with at least one translation table walk. `1` This event is implemented.
[19]	LL_CACHE_MISS	Attributable last level data or unified cache miss. `0` This event is not implemented.
[18]	LL_CACHE	Attributable last level data cache access. `0` This event is not implemented.
[17]	REMOTE_ACCESS	Attributable access to another socket in a multi-socket system. `1` This event is implemented.
[16]	L2I_TLB	Attributable L2 instruction TLB access. `0` This event is not implemented.
[15]	L2D_TLB	Attributable L2 data or unified TLB access. `1` This event is implemented.
[14]	L2I_TLB_REFILL	Attributable L2 instruction TLB refill. `0` This event is not implemented.
[13]	L2D_TLB_REFILL	Attributable L2 data or unified TLB refill. `1` This event is implemented.
[12]	L3D_CACHE_WB	Attributable L3 data or unified TLB access. `0` This event is not implemented.
[11]	L3D_CACHE	Attributable L3 data cache access. `1` This event is implemented.
[10]	L3D_CACHE_REFILL	Attributable L3 data cache refill. `1` This event is implemented.
[9]	L3D_CACHE_ALLOCATE	Attributable L3 data or unified cache allocation without refill. `1` This event is implemented.
[8]	L2I_CACHE_REFILL	Attributable L2 unified cache refill. `0` This event is not implemented.

Table D4-3 PMU common events (continued)

Bit	Event mnemonic	Description
[7]	L2I_CACHE	Attributable L2 unified cache access. `0` This event is not implemented.
[6]	L1I_TLB	Attributable L1 instruction TLB access. `1` This event is implemented.
[5]	L1D_TLB	Attributable L1 data or unified TLB access. `1` This event is implemented.
[4]	STALL_BACKEND	No operation issued due to backend. `1` This event is implemented.
[3]	STALL_FRONTEND	No operation issued due to frontend. `1` This event is implemented.
[2]	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch. `1` This event is implemented.
[1]	BR_RETIRED	Instruction architecturally executed, branch. `1` This event is implemented.
[0]	L2D_CACHE_ALLOCATE	Attributable L2 unified cache allocation without refill. `1` This event is implemented.

D4.4 PMCEID2, Performance Monitors Common Event Identification Register 2

The PMCEID2 defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

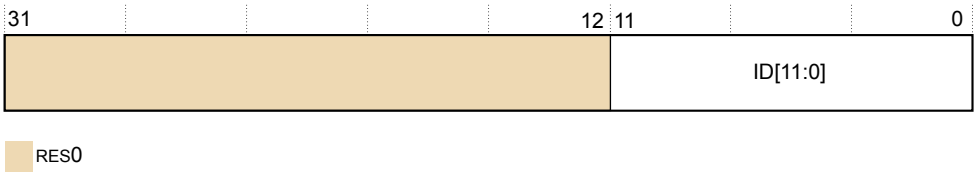

Figure D4-3 PMCEID2 bit assignments

RES0, [31:16]

RES0 Reserved

ID, [11:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

For each bit described in the following table, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

Table D4-4 PMU common events

Bit	Event mnemonic	Description
[31:12]	RES0	Reserved
[11]	L3_CACHE_LMISS_RD	L3 unified cache long-latency read miss: `1` This event is implemented.
[10]	L2I_CACHE_LMISS	L2 unified cache long-latency miss: `0` This event is not implemented.
[9]	L2D_CACHE_LMISS_RD	L2 unified cache long-latency read miss: `1` This event is implemented.
[8]	RES0	Reserved
[7]	RES0	Reserved
[6]	L1I_CACHE_LMISS	L1 instruction cache long-latency miss: `1` This event is implemented.
[5]	STALL_BACKEND_MEM	Memory stall cycles: `1` This event is implemented.
[4]	CNT_CYCLES	Constant frequency cycles: `1` This event is implemented.

Table D4-4 PMU common events (continued)

Bit	Event mnemonic	Description
[3]	SAMPLE_COLLISION	Sample collided with previous sample: 1 This event is implemented.
[2]	SAMPLE_FILTRATE	Sample taken and not removed by filtering: 1 This event is implemented.
[1]	SAMPLE_FEED	Sample taken: 1 This event is implemented.
[0]	SAMPLE_POP	Sample population: 1 This event is implemented.

D4.5 PMCR, Performance Monitors Control Register

The PMCR provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Bit field descriptions

PMCR is a 32-bit register, and is part of the Performance Monitors registers functional group.

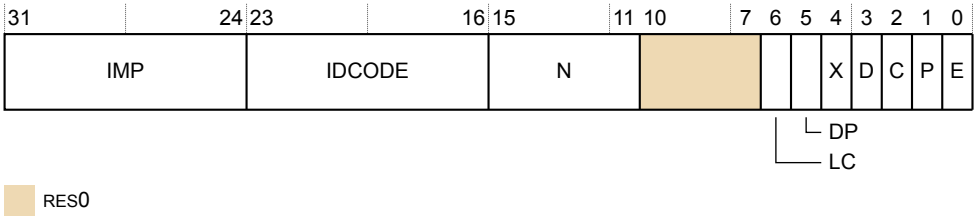

Figure D4-4 PMCR bit assignments

IMP, [31:24]

Indicates the implementer code. The value is:

`0x41` ASCII character 'A' - implementer is Arm Limited.

IDCODE, [23:16]

Identification code. The value is:

`0x21` Cortex-A78C core

N, [15:11]

Identifies the number of event counters implemented.

`0b00110` The core implements six event counters.

RES0, [10:7]

RES0 Reserved

LC, [6]

Long cycle count enable. Determines which PMCCNTR bit generates an overflow recorded in PMOVSR[31]. The overflow event is generated on a 32-bit or 64-bit boundary. The possible values are:

`0b0` Overflow event is generated on a 32-bit boundary, when an increment changes PMCCNTR[31] from 1 to 0. This is the reset value.

`0b1` Overflow event is generated on a 64-bit boundary, when an increment changes PMCCNTR[63] from 1 to 0.

Arm deprecates use of PMCR.LC = `0b0`.

DP, [5]

Disable cycle counter CCNT when event counting is prohibited. The possible values are:

`0b0` Cycle counter operates regardless of the non-invasive debug authentication settings. This is the reset value.

`0b1` Cycle counter is disabled if non-invasive debug is not permitted and enabled.

X, [4]

Export enable. This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus. The possible values are:

`0b0` Export of events is disabled. This is the reset value.

`0b1` Export of events is enabled.

No events are exported when counting is prohibited.

This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.

When this register has an architecturally defined reset value, if this field is implemented as an RW field, it resets to 0.

D, [3]

Clock divider. The possible values are:

`0b0` When enabled, counter CCNT counts every clock cycle. This is the reset value.

`0b1` When enabled, counter CCNT counts once every 64 clock cycles.

Arm deprecates use of PMCR.D = `0b1`.

C, [2]

Cycle counter reset. This bit is WO. The effects of writing to this bit are:

`0b0` No action. This is the reset value.

`0b1` Reset PMCCNTR to zero.

This bit is always RAZ.

Resetting PMCCNTR does not clear the PMCCNTR overflow bit to 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

P, [1]

Event counter reset. This bit is WO. The effects of writing to this bit are:

`0b0` No action. This is the reset value.

`0b1` Reset all event counters accessible in the current EL, not including PMCCNTR, to zero.

This bit is always RAZ.

In Non-secure EL0 and EL1, a write of 1 to this bit does not reset event counters that HDCR.HPMN or MDCR_EL2.HPMN reserves for EL2 use.

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

Resetting the event counters does not clear any overflow bits to 0.

E, [0]

Enable. The possible values are:

`0b0` All counters that are accessible at Non-secure EL1, including PMCCNTR, are disabled. This is the reset value.

`0b1` When this register has an architecturally defined reset value, this field resets to 0.

This bit is RW.

This bit does not affect the operation of event counters that HDCR.HPMN or MDCR_EL2.HPMN reserves for EL2 use.

When this register has an architecturally defined reset value, this field resets to 0.

Configurations

AArch32 System register PMCR is architecturally mapped to AArch64 System register PMCR_EL0. See *D5.4 PMCR_EL0, Performance Monitors Control Register, EL0* on page D5-517.

AArch32 System register PMCR bits [6:0] are architecturally mapped to External register PMCR_EL0[6:0].

There is one instance of this register that is used in both Secure and Non-secure states.

This register is in the Warm reset domain. Some or all RW fields of this register have defined reset values. On a Warm or Cold reset these apply only if the PE resets into an Exception level that is using AArch32. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

D4.6 PMMIR, Performance Monitors Machine Identification Register

The PMMIR defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

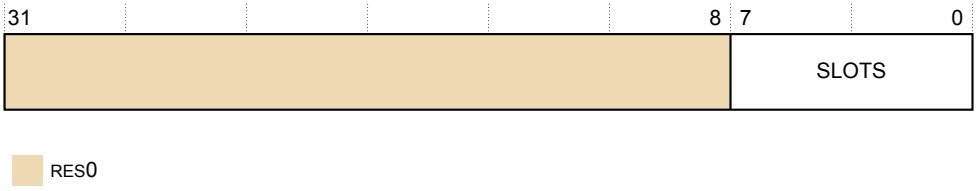

Figure D4-5 PMMIR bit assignments

RES0, [31:8]

RES0 Reserved

SLOTS, [7:0]

`0x06` Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might Read-As-Zero.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Chapter D5
AArch64 PMU registers

This chapter describes the AArch64 *Performance Monitoring Unit* (PMU) registers and shows examples of how to use them.

It contains the following sections:

D5.1 AArch64 PMU register summary

The *Performance Monitoring Unit* (PMU) counters and their associated control registers are accessible in the AArch64 Execution state with `MRS` and `MSR` instructions.

The following table gives a summary of the Cortex-A78C PMU registers in the AArch64 Execution state. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table D5-1 PMU register summary in the AArch64 Execution state

Name	Type	Width	Reset	Description
PMCR_EL0	RW	32	`0x412130XX`	*D5.4 PMCR_EL0, Performance Monitors Control Register, EL0* on page D5-517
PMCNTENSET_EL0	RW	32	UNK	Performance Monitors Count Enable Set Register
PMCNTENCLR_EL0	RW	32	UNK	Performance Monitors Count Enable Clear Register
PMOVSCLR_EL0	RW	32	UNK	Performance Monitors Overflow Flag Status Register
PMSWINC_EL0	WO	32	UNK	Performance Monitors Software Increment Register
PMSELR_EL0	RW	32	UNK	Performance Monitors Event Counter Selection Register
PMCEID0_EL0	RO	64	`0x00000A7F7FFF0F3F`	*D5.2 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0, EL0* on page D5-510
PMCEID1_EL0	RO	64	`0x00000000FEF2AE7F`	*D5.3 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1, EL0* on page D5-514
PMCCNTR_EL0	RW	64	UNK	Performance Monitors Cycle Count Register
PMXEVTYPER_EL0	RW	32	UNK	Performance Monitors Selected Event Type and Filter Register

Table D5-1 PMU register summary in the AArch64 Execution state (continued)

Name	Type	Width	Reset	Description
PMCCFILTR_EL0	RW	32	UNK	Performance Monitors Cycle Count Filter Register
PMXEVCNTR_EL0	RW	32	UNK	Performance Monitors Selected Event Count Register
PMUSERENR_EL0	RW	32	UNK	Performance Monitors User Enable Register
PMINTENSET_EL1	RW	32	UNK	Performance Monitors Interrupt Enable Set Register
PMINTENCLR_EL1	RW	32	UNK	Performance Monitors Interrupt Enable Clear Register
PMOVSSET_EL0	RW	32	UNK	Performance Monitors Overflow Flag Status Set Register
PMEVCNTR0_EL0	RW	32	UNK	Performance Monitors Event Count Registers
PMEVCNTR1_EL0	RW	32	UNK	
PMEVCNTR2_EL0	RW	32	UNK	
PMEVCNTR3_EL0	RW	32	UNK	
PMEVCNTR4_EL0	RW	32	UNK	
PMEVCNTR5_EL0	RW	32	UNK	
PMEVTYPER0_EL0	RW	32	UNK	Performance Monitors Event Type Registers
PMEVTYPER1_EL0	RW	32	UNK	
PMEVTYPER2_EL0	RW	32	UNK	
PMEVTYPER3_EL0	RW	32	UNK	
PMEVTYPER4_EL0	RW	32	UNK	
PMEVTYPER5_EL0	RW	32	UNK	
PMCCFILTR_EL0	RW	32	UNK	Performance Monitors Cycle Count Filter Register

Related references

C2.3 PMU events on page C2-428

D5.2 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0, EL0

The PMCEID0_EL0 defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

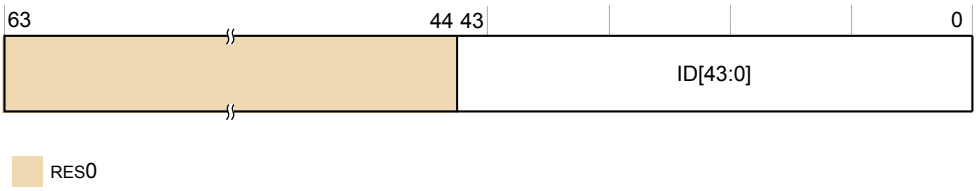

Figure D5-1 PMCEID0_EL0 bit assignments

RES0, [63:44]

RES0 Reserved

ID, [43:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

For each bit described in the following table, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

Table D5-2 PMU common events

Bit	Event mnemonic	Description
[43]	L3D_CACHE_LMISS_RD	L3 unified cache long-latency read miss: 1 This event is implemented.
[42]	L2I_CACHE_LMISS	L2 unified cache long-latency miss: 0 This event is not implemented.
[41]	L2D_CACHE_LMISS_RD	L2 unified cache long-latency read miss: 1 This event is implemented.
[40]	RES0	Reserved
[39]	RES0	Reserved
[38]	L1I_CACHE_LMISS	L1 instruction cache long-latency miss: 1 This event is implemented.
[37]	STALL_BACKEND_MEM	Memory stall cycles: 1 This event is implemented.
[36]	CNT_CYCLES	Constant frequency cycles: 1 This event is implemented.

Table D5-2 PMU common events (continued)

Bit	Event mnemonic	Description
[35]	SAMPLE_COLLISION	Sample collided with previous sample: 1 This event is implemented.
[34]	SAMPLE_FILTRATE	Sample taken, not removed: 1 This event is implemented.
[33]	SAMPLE_FEED	Sample taken: 1 This event is implemented.
[32]	SAMPLE_POP	Sample population: 1 This event is implemented.
[31]	L1D_CACHE_ALLOCATE	L1 Data cache allocate: 0 This event is not implemented.
[30]	CHAIN	Chain. For odd-numbered counters, counts once for each overflow of the preceding even-numbered counter. For even-numbered counters, does not count: 1 This event is implemented.
[29]	BUS_CYCLES	Bus cycle: 1 This event is implemented.
[28]	TTBR_WRITE_RETIRED	TTBR write, architecturally executed, condition check pass - write to translation table base: 1 This event is implemented.
[27]	INST_SPEC	Instruction speculatively executed: 1 This event is implemented.
[26]	MEMORY_ERROR	Local memory error: 1 This event is implemented.
[25]	BUS_ACCESS	Bus access: 1 This event is implemented.
[24]	L2D_CACHE_WB	L2 unified cache Write-Back: 1 This event is implemented.
[23]	L2D_CACHE_REFILL	L2 unified cache refill: 1 This event is implemented.
[22]	L2D_CACHE	L2 unified cache access: 1 This event is implemented.

Table D5-2 PMU common events (continued)

Bit	Event mnemonic	Description
[21]	L1D_CACHE_WB	L1 Data cache Write-Back: `1` This event is implemented.
[20]	L1I_CACHE	L1 Instruction cache access: `1` This event is implemented.
[19]	MEM_ACCESS	Data memory access: `1` This event is implemented.
[18]	BR_PRED	Predictable branch speculatively executed: `1` This event is implemented.
[17]	CPU_CYCLES	Cycle: `1` This event is implemented.
[16]	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed: `1` This event is implemented.
[15]	UNALIGNED_LDST_RETIRED	Instruction architecturally executed, condition check pass - unaligned load or store: `0` This event is not implemented.
[14]	BR_RETURN_RETIRED	Instruction architecturally executed, condition check pass - procedure return: `0` This event is not implemented.
[13]	BR_IMMED_RETIRED	Instruction architecturally executed - immediate branch: `0` This event is not implemented.
[12]	PC_WRITE_RETIRED	Instruction architecturally executed, condition check pass - software change of the PC: `0` This event is not implemented.
[11]	CID_WRITE_RETIRED	Instruction architecturally executed, condition check pass - write to CONTEXTIDR: `1` This event is implemented.
[10]	EXC_RETURN	Instruction architecturally executed, condition check pass - exception return: `1` This event is implemented.
[9]	EXC_TAKEN	Exception taken: `1` This event is implemented.
[8]	INST_RETIRED	Instruction architecturally executed: `1` This event is implemented.

Table D5-2 PMU common events (continued)

Bit	Event mnemonic	Description
[7]	ST_RETIRED	Instruction architecturally executed, condition check pass - store: `0` This event is not implemented.
[6]	LD_RETIRED	Instruction architecturally executed, condition check pass - load: `0` This event is not implemented.
[5]	L1D_TLB_REFILL	L1 Data TLB refill: `1` This event is implemented.
[4]	L1D_CACHE	L1 Data cache access: `1` This event is implemented.
[3]	L1D_CACHE_REFILL	L1 Data cache refill: `1` This event is implemented.
[2]	L1I_TLB_REFILL	L1 Instruction TLB refill: `1` This event is implemented.
[1]	L1I_CACHE_REFILL	L1 Instruction cache refill: `1` This event is implemented.
[0]	SW_INCR	Instruction architecturally executed, condition check pass - software increment: `1` This event is implemented.

Note

The PMU events implemented in the above table can be found in *Table C2-1 PMU Events on page C2-428*.

D5.3 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1, EL0

The PMCEID1_EL0 defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

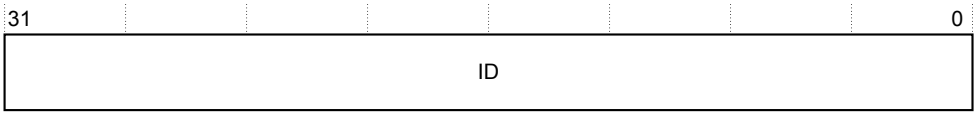

Figure D5-2 PMCEID1_EL0 bit assignments

RES0, [63:32]

RES0 Reserved

ID, [31:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

For each bit described in the following table, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

Table D5-3 PMU common events

Bit	Event mnemonic	Description
[63:32]	RES0	Reserved
[31]	STALL_SLOT	No operation sent for execution on a slot: 1 This event is implemented.
[30]	STALL_SLOT_FRONTEND	No operation sent for execution on a slot due to the frontend: 1 This event is implemented.
[29]	STALL_SLOT_BACKEND	No operation sent for execution on a slot due to the backend: 1 This event is implemented.
[28]	STALL	No operation sent for execution: 1 This event is implemented.
[27]	OP_SPEC	Micro-operation speculatively executed: 1 This event is implemented.
[26]	OP_RETIRED	Micro-operation architecturally executed: 1 This event is implemented.
[25]	L1D_CACHE_LMISS_RD	L1 data cache long-latency read miss: 1 This event is implemented.

Table D5-3 PMU common events (continued)

Bit	Event mnemonic	Description
[24]	REMOTE_ACCESS_RD	Attributable memory read access to another socket in a multi-socket system: 0 This event is not implemented.
[23]	LL_CACHE_MISS_RD	Attributable last level cache memory read miss: 1 This event is implemented.
[22]	LL_CACHE_RD	Attributable last level cache memory read: 1 This event is implemented.
[21]	ITLB_WALK	Attributable instruction TLB access with at least one translation table walk: 1 This event is implemented.
[20]	DTLB_WALK	Attributable data or unified TLB access with at least one translation table walk: 1 This event is implemented.
[19]	LL_CACHE_MISS	Attributable last level data or unified cache miss: 0 This event is not implemented.
[18]	LL_CACHE	Attributable last level data cache access: 0 This event is not implemented.
[17]	REMOTE_ACCESS	Attributable access to another socket in a multi-socket system: 1 This event is implemented.
[16]	L2I_TLB	Attributable L2 unified TLB access: 0 This event is not implemented.
[15]	L2D_TLB	Attributable L2 unified TLB access: 1 This event is implemented.
[14]	L2I_TLB_REFILL	Attributable L2 unified TLB refill: 0 This event is not implemented.
[13]	L2D_TLB_REFILL	Attributable L2 data unified TLB refill: 1 This event is implemented.
[12]	L3D_CACHE_WB	Attributable L3 unified TLB access: 0 This event is not implemented.
[11]	L3D_CACHE	Attributable L3 unified cache access: 1 This event is implemented.
[10]	L3D_CACHE_REFILL	Attributable L3 unified cache refill: 1 This event is implemented.

Table D5-3 PMU common events (continued)

Bit	Event mnemonic	Description
[9]	L3D_CACHE_ALLOCATE	Attributable L3 unified cache allocation without refill: `1` This event is implemented.
[8]	L2I_CACHE_REFILL	Attributable L2 unified cache refill: `0` This event is not implemented.
[7]	L2I_CACHE	Attributable L2 unified cache access: `0` This event is not implemented.
[6]	L1I_TLB	Attributable L1 instruction TLB access: `1` This event is implemented.
[5]	L1D_TLB	Attributable L1 data or unified TLB access: `1` This event is implemented.
[4]	STALL_BACKEND	No operation issued due to backend: `1` This event is implemented.
[3]	STALL_FRONTEND	No operation issued due to frontend: `1` This event is implemented.
[2]	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch: `1` This event is implemented.
[1]	BR_RETIRED	Instruction architecturally executed, branch: `1` This event is implemented.
[0]	L2D_CACHE_ALLOCATE	Attributable L2 unified cache allocation without refill: `1` This event is implemented.

D5.4 PMCR_EL0, Performance Monitors Control Register, EL0

The PMCR_EL0 provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Bit field descriptions

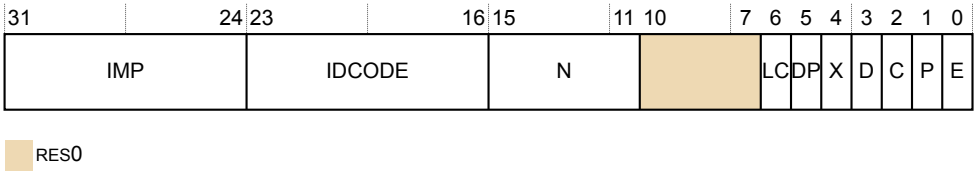

Figure D5-3 PMCR_EL0 bit assignments

IMP, [31:24]

Implementer code:

`0x41` Arm

This is a read-only field.

IDCODE, [23:16]

Identification code:

`0x24` Cortex-A78C

This is a read-only field.

N, [15:11]

Number of event counters

`0b00110` Six counters

RES0, [10:7]

RES0 Reserved

LC, [6]

Long cycle count enable. Determines which PMCCNTR_EL0 bit generates an overflow recorded in PMOVSR[31]. The possible values are:

`0` Overflow on increment that changes PMCCNTR_EL0[31] from 1 to 0

`1` Overflow on increment that changes PMCCNTR_EL0[63] from 1 to 0

DP, [5]

Disable cycle counter, PMCCNTR_EL0 when event counting is prohibited:

`0` Cycle counter operates regardless of the non-invasive debug authentication settings. This is the reset value.

`1` Cycle counter is disabled if non-invasive debug is not permitted and enabled.

This bit is read/write.

X, [4]

Export enable. This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus:

`0` Export of events is disabled. This is the reset value.

`1` Export of events is enabled.

This bit is read/write and does not affect the generation of Performance Monitors interrupts on the **nPMUIRQ** pin.

D, [3]

Clock divider:

`0` When enabled, PMCCNTR_EL0 counts every clock cycle. This is the reset value.

`1` When enabled, PMCCNTR_EL0 counts every 64 clock cycles.

This bit is read/write.

C, [2]

Clock counter reset. This bit is WO. The effects of writing to this bit are:

`0` No action. This is the reset value.

`1` Reset PMCCNTR_EL0 to 0.

This bit is always RAZ.

Resetting PMCCNTR_EL0 does not clear the PMCCNTR_EL0 overflow bit to 0. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

P, [1]

Event counter reset. This bit is WO. The effects of writing to this bit are:

`0` No action. This is the reset value.

`1` Reset all event counters, not including PMCCNTR_EL0, to zero.

This bit is always RAZ.

In Non-secure EL0 and EL1, a write of 1 to this bit does not reset event counters that MDCR_EL2.HPMN reserves for EL2 use.

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

Resetting the event counters does not clear any overflow bits to 0.

E, [0]

Enable. The possible values of this bit are:

`0` All counters, including PMCCNTR_EL0, are disabled. This is the reset value.

`1` All counters are enabled.

This bit is RW.

In Non-secure EL0 and EL1, this bit does not affect the operation of event counters that MDCR_EL2.HPMN reserves for EL2 use.

On Warm reset, the field resets to 0.

Configurations

AArch64 System register PMCR_EL0 is architecturally mapped to AArch32 System register PMCR.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

D5.5 CPUPMMIR_EL1, Performance Monitors Machine Identification Register, EL1

The CPUPMMIR_EL1 is a 64-bit register that defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

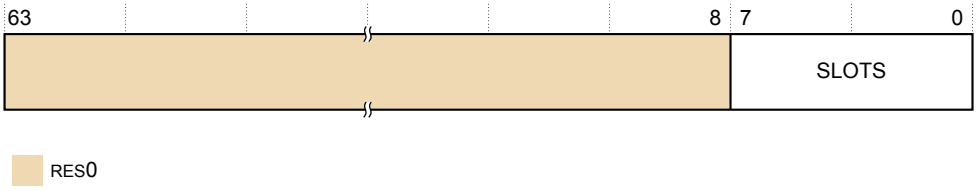

Figure D5-4 CPUPMMIR_EL1 bit assignments

RES0, [31:8]

RES0 Reserved

SLOTS, [7:0]

0x06 Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero.

Usage constraints

Accessing the CPUPMMIR_EL1

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This syntax is encoded with the following settings in the instruction encoding:

<systemreg>	op0	CRn	op1	op2	CRm
PMMIR_EL1	3	15	0	14	6

Chapter D6
Memory-mapped PMU registers

This chapter describes the memory-mapped *Performance Monitoring Unit* (PMU) registers and shows examples of how to use them.

It contains the following sections:

D6.1 Memory-mapped PMU register summary

There are *Performance Monitoring Unit* (PMU) registers that are accessible through the external debug interface.

These registers are listed in the following table. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table D6-1 Memory-mapped PMU register summary

Offset	Name	Type	Description
0x000	PMEVCNTR0_EL0	RW	Performance Monitor Event Count Register 0
0x004	-	-	Reserved
0x008	PMEVCNTR1_EL0	RW	Performance Monitor Event Count Register 1
0x00C	-	-	Reserved
0x010	PMEVCNTR2_EL0	RW	Performance Monitor Event Count Register 2
0x014	-	-	Reserved
0x018	PMEVCNTR3_EL0	RW	Performance Monitor Event Count Register 3
0x01C	-	-	Reserved
0x020	PMEVCNTR4_EL0	RW	Performance Monitor Event Count Register 4
0x024	-	-	Reserved
0x028	PMEVCNTR5_EL0	RW	Performance Monitor Event Count Register 5
0x02C-0x0F4	-	-	Reserved
0x0F8	PMCCNTR_EL0[31:0]	RW	Performance Monitor Cycle Count Register
0x0FC	PMCCNTR_EL0[63:32]	RW	
0x100-0x1FC	-	-	Reserved
0x200	PMPCSR[31:0]	RO	Program Counter Sample Register
0x204	PMPCSR[63:32]		
0x208	PMCID1SR	RO	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	RO	VMID Sample Register
0x220	PMPCSR[31:0]	RO	Program Counter Sample Register (alias)
0x224	PMPCSR[63:32]		
0x228	PMCID1SR	RO	CONTEXTIDR_EL1 Sample Register (alias)

Table D6-1 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
0x22C	PMCID2SR	RO	CONTEXTIDR_EL2 Sample Register
0x230-0x3FC	-	-	Reserved
0x400	PMEVTYPER0_EL0	RW	Performance Monitors Event Type Register 0
0x404	PMEVTYPER1_EL0	RW	Performance Monitors Event Type Register 1
0x408	PMEVTYPER2_EL0	RW	Performance Monitors Event Type Register 2
0x40C	PMEVTYPER3_EL0	RW	Performance Monitors Event Type Register 3
0x410	PMEVTYPER4_EL0	RW	Performance Monitors Event Type Register 4
0x414	PMEVTYPER5_EL0	RW	Performance Monitors Event Type Register 5
0x418-0x478	-	-	Reserved
0x47C	PMCCFILTR_EL0	RW	Performance Monitors Cycle Count Filter Register
0xC00	PMCNTENSET_EL0	RW	Performance Monitor Count Enable Set Register
0xC04-0xC1C	-	-	Reserved
0xC20	PMCNTENCLR_EL0	RW	Performance Monitor Count Enable Clear Register
0xC24-0xC3C	-	-	Reserved
0xC40	PMINTENSET_EL1	RW	Performance Monitor Interrupt Enable Set Register
0xC44-0xC5C	-	-	Reserved
0xC60	PMINTENCLR_EL1	RW	Performance Monitor Interrupt Enable Clear Register
0xC64-0xC7C	-	-	Reserved
0xC80	PMOVSCLR_EL0	RW	Performance Monitor Overflow Flag Status Register
0xC84-0xC9C	-	-	Reserved
0xCA0	PMSWINC_EL0	WO	Performance Monitor Software Increment Register
0xCA4-0xCBC	-	-	Reserved
0xCC0	PMOVSSET_EL0	RW	Performance Monitor Overflow Flag Status Set Register
0xCC4-0xDFC	-	-	Reserved

Table D6-1 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
0xE00	PMCFGR	RO	*D6.2 PMCFGR, Performance Monitors Configuration Register* on page D6-526
0xE04	PMCR_EL0	RW	Performance Monitors Control Register. This register is distinct from the PMCR_EL0 system register. It does not have the same value.
0xE08-0xE1C	-	-	Reserved
0xE20	PMCEID0	RO	*D4.2 PMCEID0, Performance Monitors Common Event Identification Register 0* on page D4-494
0xE24	PMCEID1	RO	*D4.3 PMCEID1, Performance Monitors Common Event Identification Register 1* on page D4-497
0xE28	PMCEID2	RO	*D4.4 PMCEID2, Performance Monitors Common Event Identification Register 2* on page D4-500
0xE2C	PMCEID3	RO	Performance Monitors Common Event Identification Register 3
0xE40	PMMIR	RO	*D6.7 PMMIR, Performance Monitors Machine Identification Register* on page D6-531
0xFA4	-	-	Reserved
0xFA8	PMDEVAFF0	RO	*B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314
0xFAC	PMDEVAFF1	RO	*B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314
0xFB8	PMAUTHSTATUS	RO	Performance Monitor Authentication Status Register
0xFBC	PMDEVARCH	RO	Performance Monitor Device Architecture Register
0xFC0-0xFC8	-	-	Reserved
0xFCC	PMDEVTYPE	RO	Performance Monitor Device Type Register
0xFD0	PMPIDR4	RO	*D6.12 PMPIDR4, Performance Monitors Peripheral Identification Register 4* on page D6-536
0xFD4	PMPIDR5	RO	*D6.13 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7* on page D6-537
0xFD8	PMPIDR6	RO	
0xFDC	PMPIDR7	RO	

Table D6-1 Memory-mapped PMU register summary (continued)

Offset	Name	Type	Description
0xFE0	PMPIDR0	RO	*D6.8 PMPIDR0, Performance Monitors Peripheral Identification Register 0* on page D6-532
0xFE4	PMPIDR1	RO	*D6.9 PMPIDR1, Performance Monitors Peripheral Identification Register 1* on page D6-533
0xFE8	PMPIDR2	RO	*D6.10 PMPIDR2, Performance Monitors Peripheral Identification Register 2* on page D6-534
0xFEC	PMPIDR3	RO	*D6.11 PMPIDR3, Performance Monitors Peripheral Identification Register 3* on page D6-535
0xFF0	PMCIDR0	RO	*D6.3 PMCIDR0, Performance Monitors Component Identification Register 0* on page D6-527
0xFF4	PMCIDR1	RO	*D6.4 PMCIDR1, Performance Monitors Component Identification Register 1* on page D6-528
0xFF8	PMCIDR2	RO	*D6.5 PMCIDR2, Performance Monitors Component Identification Register 2* on page D6-529
0xFFC	PMCIDR3	RO	*D6.6 PMCIDR3, Performance Monitors Component Identification Register 3* on page D6-530

D6.2 PMCFGR, Performance Monitors Configuration Register

The PMCFGR contains *Performance Monitoring Unit* (PMU) specific configuration data.

Bit field descriptions

The PMCFGR is a 32-bit register.

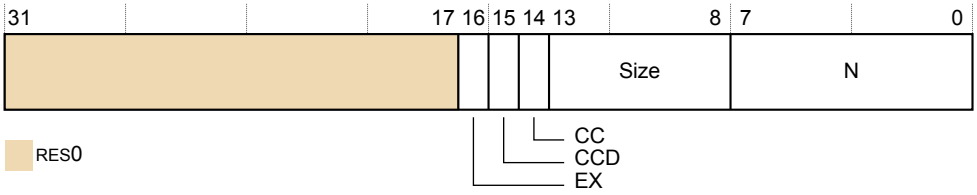

Figure D6-1 PMCFGR bit assignments

RES0, [31:17]

RES0 Reserved

EX, [16]

Export supported. The value is:

`1` Export is supported. PMCR_EL0.EX is read/write.

CCD, [15]

Cycle counter has pre-scale. The value is:

`1` PMCR_EL0.D is read/write.

CC, [14]

Dedicated cycle counter supported. The value is:

`1` Dedicated cycle counter is supported.

Size, [13:8]

Counter size. The value is:

`0b111111` 64-bit counters

N, [7:0]

Number of event counters. The value is:

`0x06` Six counters

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMCFGR can be accessed through the external debug interface, offset `0xE00`.

D6.3 PMCIDR0, Performance Monitors Component Identification Register 0

The PMCIDR0 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMCIDR0 is a 32-bit register.

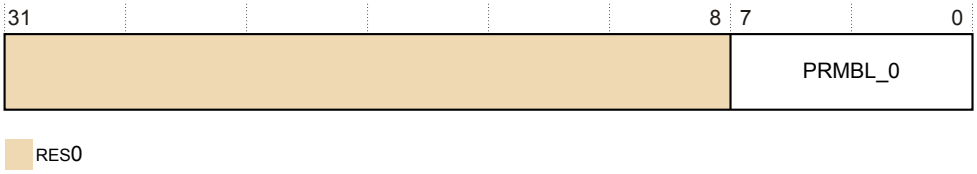

Figure D6-2 PMCIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_0, [7:0]

`0x0D` Preamble byte 0

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

D6.4 PMCIDR1, Performance Monitors Component Identification Register 1

The PMCIDR1 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMCIDR1 is a 32-bit register.

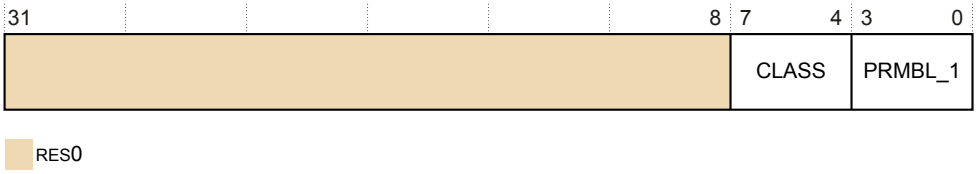

Figure D6-3 PMCIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

CLASS, [7:4]

`0x9` Debug component

PRMBL_1, [3:0]

`0x0` Preamble byte 1

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

D6.5 PMCIDR2, Performance Monitors Component Identification Register 2

The PMCIDR2 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMCIDR2 is a 32-bit register.

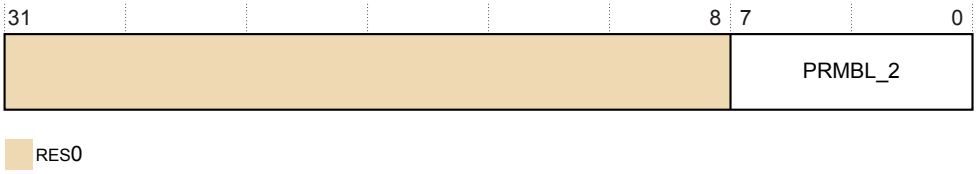

Figure D6-4 PMCIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_2, [7:0]

`0x05` Preamble byte 2

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

D6.6 PMCIDR3, Performance Monitors Component Identification Register 3

The PMCIDR3 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMCIDR3 is a 32-bit register.

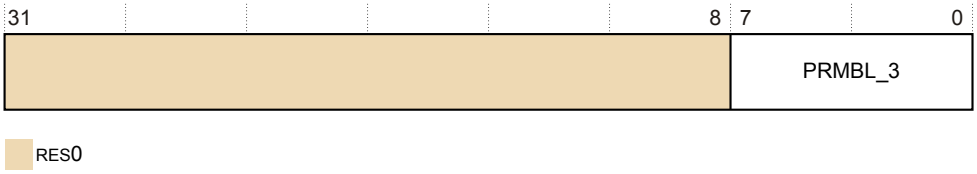

Figure D6-5 PMCIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_3, [7:0]

`0xB1` Preamble byte 3

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

D6.7 PMMIR, Performance Monitors Machine Identification Register

The PMMIR defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

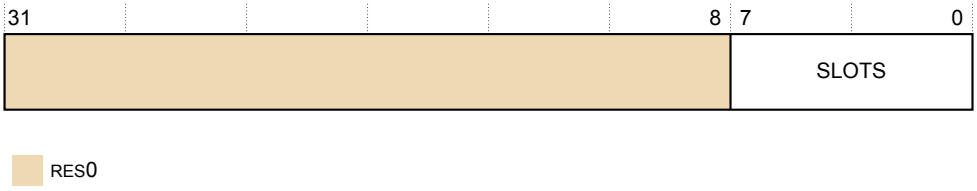

Figure D6-6 PMMIR bit assignments

RES0, [31:8]

RES0 Reserved

SLOTS, [7:0]

`0x06` Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero.

The PMMIR can be accessed through the external debug interface, offset `0xD80`.

D6.8 PMPIDR0, Performance Monitors Peripheral Identification Register 0

The PMPIDR0 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMPIDR0 is a 32-bit register.

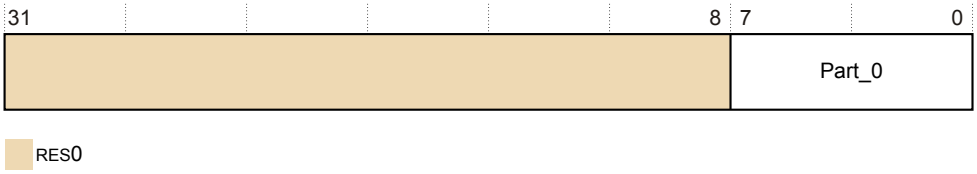

Figure D6-7 PMPIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

Part_0, [7:0]

`0x4B` Least significant byte of the performance monitor part number

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

D6.9 PMPIDR1, Performance Monitors Peripheral Identification Register 1

The PMPIDR1 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMPIDR1 is a 32-bit register.

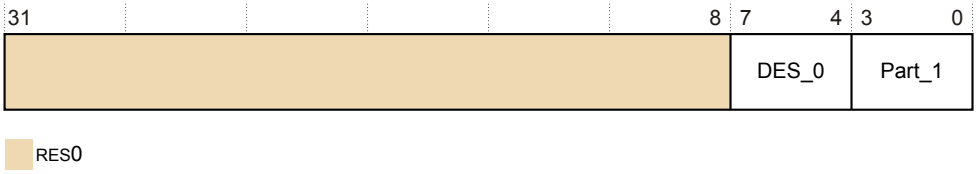

Figure D6-8 PMPIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

DES_0, [7:4]

`0xB` Arm Limited. This is the least significant nibble of JEP106 ID code.

Part_1, [3:0]

`0xD` Most significant nibble of the performance monitor part number

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

D6.10 PMPIDR2, Performance Monitors Peripheral Identification Register 2

The PMPIDR2 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMPIDR2 is a 32-bit register.

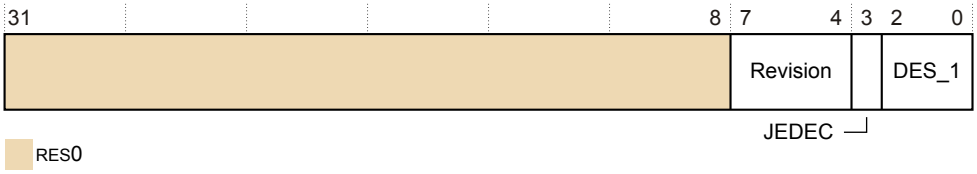

Figure D6-9 PMPIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

Revision, [7:4]

`0x1` r0p1

JEDEC, [3]

`0b1` RAO. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is the most significant nibble of JEP106 ID code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

D6.11 PMPIDR3, Performance Monitors Peripheral Identification Register 3

The PMPIDR3 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMPIDR3 is a 32-bit register.

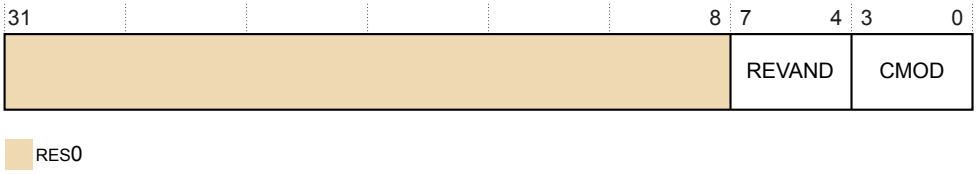

Figure D6-10 PMPIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

REVAND, [7:4]

`0x0` Part minor revision

CMOD, [3:0]

`0x0` Customer modified

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

D6.12 PMPIDR4, Performance Monitors Peripheral Identification Register 4

The PMPIDR4 provides information to identify a Performance Monitor component.

Bit field descriptions

The PMPIDR4 is a 32-bit register.

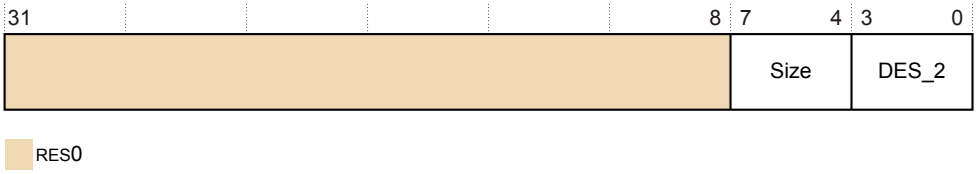

Figure D6-11 PMPIDR4 bit assignments

RES0, [31:8]

RES0 Reserved

Size, [7:4]

`0x0` Size of the component. Log_2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited. This is the least significant nibble JEP106 continuation code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

D6.13 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

Chapter D7
PMU snapshot registers

Performance Monitoring Unit (PMU) snapshot registers are an IMPLEMENTATION DEFINED extension to an Armv8-A compliant PMU to support an external core monitor that connects to a system profiler.

It contains the following sections:

D7.1 PMU snapshot register summary

The snapshot registers are visible in an IMPLEMENTATION DEFINED region of the *Performance Monitoring Unit* (PMU) external debug interface. Each time the debugger sends a snapshot request, information is collected to see how the code is executed in the different cores.

The following table describes the PMU snapshot registers implemented in the core.

Table D7-1 PMU snapshot register summary

Offset	Name	Type	Width	Description
0x600	PMPCSSR_LO	RO	32	*D7.2 PMPCSSR, PMU Snapshot Program Counter Sample Register* on page D7-541
0x604	PMPCSSR_HI	RO	32	
0x608	PMCIDSSR	RO	32	*D7.3 PMCIDSSR, PMU Snapshot CONTEXTIDR_EL1 Sample Register* on page D7-542
0x60C	PMCID2SSR	RO	32	*D7.4 PMCID2SSR, PMU Snapshot CONTEXTIDR_EL2 Sample Register* on page D7-543
0x610	PMSSSR	RO	32	*D7.5 PMSSSR, PMU Snapshot Status Register* on page D7-544
0x614	PMOVSSR	RO	32	*D7.6 PMOVSSR, PMU Snapshot Overflow Status Register* on page D7-545
0x618	PMCCNTSR_LO	RO	32	*D7.7 PMCCNTSR, PMU Snapshot Cycle Counter Register* on page D7-546
0x61C	PMCCNTSR_HI	RO	32	
0x620 + 4×n	PMEVCNTSRn	RO	32	*D7.8 PMEVCNTSRn, PMU Snapshot Cycle Counter Registers 0-5* on page D7-547
0x6F0	PMSSCR	WO	32	*D7.9 PMSSCR, PMU Snapshot Capture Register* on page D7-548

D7.2 PMPCSSR, PMU Snapshot Program Counter Sample Register

The PMPCSSR holds the same value as the PMPCSR register.

However, unlike the other view of PMPCSR, it is not sensitive to reads. That is, reads of PMPCSSR through the PMU snapshot view do not cause a new sample capture and do not change PMCID1SR, PMCID2SR, or PMVIDSR.

Bit field descriptions

The PMPCSSR is a 64-bit read-only register.

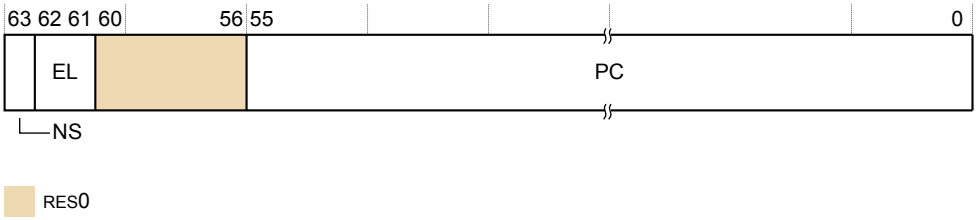

Figure D7-1 PMPCSSR bit assignments

NS, [63]
Non-secure sample

EL, [62:61]
Exception level sample

RES0, [60:56]

RES0 Reserved

PC, [55:0]
Sampled PC

Configurations
There are no configuration notes.

Usage constraints

Any access to PMPCSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

D7.3 PMCIDSSR, PMU Snapshot CONTEXTIDR_EL1 Sample Register

The PMCIDSSR holds the same value as the PMCID1SR register.

Configurations

There are no configuration notes.

Usage constraints

Any access to PMCIDSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

D7.4 PMCID2SSR, PMU Snapshot CONTEXTIDR_EL2 Sample Register

The PMCID2SSR holds the same value as the PMCID2SR register.

Configurations

There are no configuration notes.

Usage constraints

Any access to PMCID2SSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

D7.5 PMSSSR, PMU Snapshot Status Register

The PMSSSR holds status information about the captured counters.

Bit field descriptions

The PMSSSR is a 32-bit read-only register.

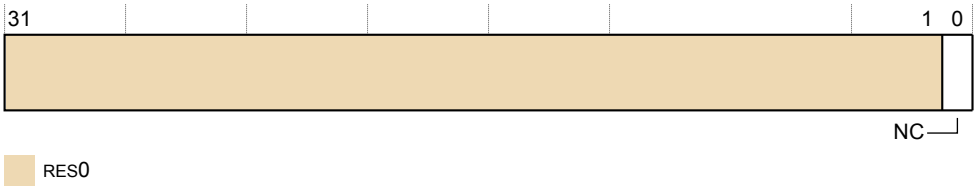

Figure D7-2 PMSSSR bit assignments

RES0, [31:1]

RES0 Reserved

NC, [0]

No capture. This bit indicates whether the PMU counters have been captured. The possible values are:

0 PMU counters are captured.

1 PMU counters are not captured.

If there is a security violation, the core does not capture the event counters. The external monitor is responsible for keeping track of whether it managed to capture the snapshot registers from the core.

This bit does not reflect the status of the captured Program Counter Sample registers.

The core resets this bit to 1 by a Warm reset but MPSSSR.NC is overwritten at the first capture.

Configurations

There are no configuration notes.

Usage constraints

Any access to PMSSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

D7.6 PMOVSSR, PMU Snapshot Overflow Status Register

The PMOVSSR is a captured copy of PMOVSR.

Once it is captured, the value in PMOVSSR is unaffected by writes to PMOVSSET_EL0 and PMOVSCLR_EL0.

Configurations

There are no configuration notes.

Usage constraints

Any access to PMOVSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

D7.7 PMCCNTSR, PMU Snapshot Cycle Counter Register

The PMCCNTSR is a captured copy of PMCCNTR_EL0.

Once it is captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR_EL0 and PMCR_EL0.C.

Configurations

There are no configuration notes.

Usage constraints

Any access to PMCCNTSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

D7.8 PMEVCNTSRn, PMU Snapshot Cycle Counter Registers 0-5

The PMEVCNTSRn, are captured copies of PMEVCNTRn_EL0, n is 0-5.

When they are captured, the value in PMSSEVCNTRn is unaffected by writes to PMSSEVCNTRn_EL0 and PMCR_EL0.P.

Configurations

There are no configuration notes.

Usage constraints

Any access to PMSSEVCNTRn returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

D7.9 PMSSCR, PMU Snapshot Capture Register

The PMSSCR provides a mechanism for software to initiate a sample.

Bit field descriptions

The PMSSCR is a 32-bit write-only register.

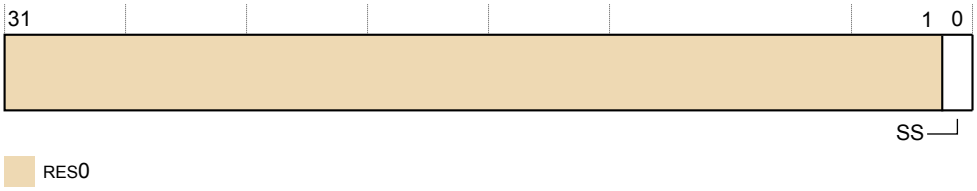

Figure D7-3 PMSSCR bit assignments

RES0, [31:1]

RES0 Reserved

SS, [0]

Capture now. The possible values are:

0 IGNORED

1 Initiate a capture immediately

Configurations

There are no configuration notes.

Usage constraints

Any access to PMSSCR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.

Chapter D8
AArch64 AMU registers

This chapter describes the AArch64 *Activity Monitor Unit* (AMU) registers and shows examples of how to use them.

It contains the following sections:

D8.1 AArch64 AMU register summary

The following table gives a summary of the Cortex-A78C *Activity Monitor Unit* (AMU) registers in the AArch64 Execution state.

Table D8-1 AArch64 AMU registers

Name	Width	Reset	Description
AMCFGR_EL0	32	0x11003F06	*D8.2 AMCFGR_EL0, Activity Monitors Configuration Register, EL0* on page D8-552
AMCGCR_EL0	32	0x00000304	*D8.3 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register, EL0* on page D8-554
AMCNTENCLR0_EL0	32	0x00000000	*D8.4 AMCNTENCLR0_EL0, Activity Monitors Count Enable Clear Register 0, EL0* on page D8-556
AMCNTENCLR1_EL0	32	0x00000000	*D8.5 AMCNTENCLR1_EL0, Activity Monitors Count Enable Clear Register 1, EL0* on page D8-558
AMCNTENSET0_EL0	32	0x00000000	*D8.6 AMCNTENSET0_EL0, Activity Monitors Count Enable Set Register 0, EL0* on page D8-560
AMCNTENSET1_EL0	32	0x00000000	*D8.7 AMCNTENSET1_EL0, Activity Monitors Count Enable Set Register 1, EL0* on page D8-562
AMCR_EL0	32	0x00000000	*D8.8 AMCR_EL0, Activity Monitors Control Register, EL0* on page D8-564
AMEVCNTR00_EL0	64	0x0000000000000000	*D8.9 AMEVCNTR0n_EL0, Activity Monitors Event Counter Registers 0n, EL0* on page D8-566
AMEVCNTR01_EL0	64	0x0000000000000000	*D8.9 AMEVCNTR0n_EL0, Activity Monitors Event Counter Registers 0n, EL0* on page D8-566
AMEVCNTR02_EL0	64	0x0000000000000000	*D8.9 AMEVCNTR0n_EL0, Activity Monitors Event Counter Registers 0n, EL0* on page D8-566
AMEVCNTR03_EL0	64	0x0000000000000000	*D8.9 AMEVCNTR0n_EL0, Activity Monitors Event Counter Registers 0n, EL0* on page D8-566
AMEVCNTR10_EL0	64	0x0000000000000000	*D8.10 AMEVCNTR1n_EL0, Activity Monitors Event Counter Registers 1n, EL0* on page D8-568
AMEVCNTR11_EL0	64	0x0000000000000000	*D8.10 AMEVCNTR1n_EL0, Activity Monitors Event Counter Registers 1n, EL0* on page D8-568
AMEVCNTR12_EL0	64	0x0000000000000000	*D8.10 AMEVCNTR1n_EL0, Activity Monitors Event Counter Registers 1n, EL0* on page D8-568
AMEVTYPER00_EL0	32	0x00000011	*D8.11 AMEVTYPER0n_EL0, Activity Monitors Event Type Registers 0n, EL0* on page D8-570
AMEVTYPER01_EL0	32	0x00004004	*D8.11 AMEVTYPER0n_EL0, Activity Monitors Event Type Registers 0n, EL0* on page D8-570
AMEVTYPER02_EL0	32	0x00000008	*D8.11 AMEVTYPER0n_EL0, Activity Monitors Event Type Registers 0n, EL0* on page D8-570
AMEVTYPER03_EL0	32	0x00004005	*D8.11 AMEVTYPER0n_EL0, Activity Monitors Event Type Registers 0n, EL0* on page D8-570
AMEVTYPER10_EL0	32	0x00000300	*D8.12 AMEVTYPER1n_EL0, Activity Monitors Event Type Registers 1n, EL0* on page D8-572

Table D8-1 AArch64 AMU registers (continued)

Name	Width	Reset	Description
AMEVTYPER11_EL0	32	0x00000301	*D8.12 AMEVTYPER1n_EL0, Activity Monitors Event Type Registers 1n, EL0* on page D8-572
AMEVTYPER12_EL0	32	0x00000302	*D8.12 AMEVTYPER1n_EL0, Activity Monitors Event Type Registers 1n, EL0* on page D8-572
AMUSERENR_EL0	32	0x00000000	*D8.13 AMUSERENR_EL0, Activity Monitors User Enable Register, EL0* on page D8-574

D8.2 AMCFGR_EL0, Activity Monitors Configuration Register, EL0

AMCFGR_EL0 provides information on the number of activity counters implemented and their size.

Bit field descriptions

AMCFGR_EL0 is a 32-bit register.

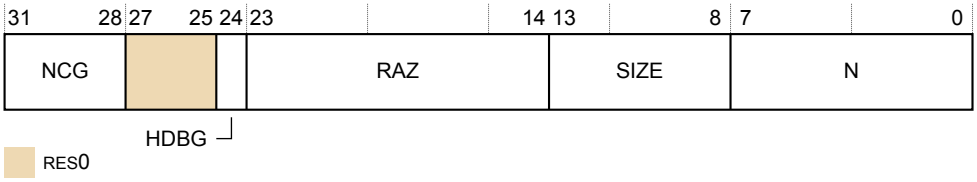

Figure D8-1 AMCFGR_EL0 bit assignments

NCG, [31:28]

Number of counter groups

The number of counter groups implemented is NCG+1. If the number of auxiliary counters implemented is zero, this field is `0x1`.

RES0, [27:25]

RES0 Reserved

HDBG, [24]

Halt-on-debug feature support

This field is required, therefore the value of this field is `0b1`.

RAZ, [23:14]

Read-As-Zero

SIZE, [13:8]

Size of counters, minus one

This field defines the size of the largest counter implemented by the activity monitors. In the Armv8-A architecture, the largest counter has 64 bits, therefore the value of this field is `0b111111`.

N, [7:0]

Number of activity counters implemented, where the number of counters is N+1. The Cortex-A78C core implements four counters, therefore the value is `0x06`.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMCFGR_EL0

To access AMCFGR_EL0:

```
MRS <Xt>, AMCFGR_EL0 ; Read AMCFGR_EL0 into Xt
```

Register access is encoded as follows:

Table D8-2 AMCFGR_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c2	1

AMCFGR_EL0 can be accessed through the external debug interface, offset `0xE00`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both Security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

  `0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps Non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.3 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register, EL0

AMCGCR_EL0 provides information on the number of activity counters implemented within each group.

Bit field descriptions

AMCGCR_EL0 is a 32-bit register.

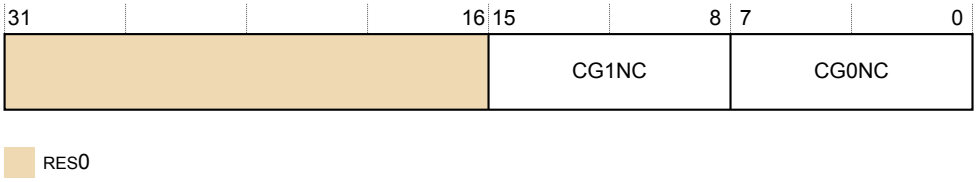

Figure D8-2 AMCGCR_EL0 bit assignments

RES0, [31:16]

RES0 Reserved

CG1NC, [15:8]

The number of counters in the auxiliary counter group. A value of `0b0001` is equal to one counter. For AMUv1 the permitted range is 0 to 16. The Cortex-A78C core implements three auxiliary counters so the value is `0x3`.

CG0NC, [7:0]

The number of counters in the architected counter group. A value of `0b0001` is equal to one counter. For AMUv1 this value is 4.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMCGCR_EL0

To access AMCGCR_EL0:

```
MRS <Xt>, AMCGCR_EL0 ; Read AMCGCR_EL0 into Xt
```

Register access is encoded as follows:

Table D8-3 AMCGCR_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c2	2

AMCGCR_EL0 can be accessed through the external debug interface, offset `0xCE0`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

  `0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.4 AMCNTENCLR0_EL0, Activity Monitors Count Enable Clear Register 0, EL0

AMCNTENCLR0_EL0 disables the activity monitor counters implemented, AMEVCNTR<0-3>_EL0.

Bit field descriptions

AMCNTENCLR0_EL0 is a 32-bit register.

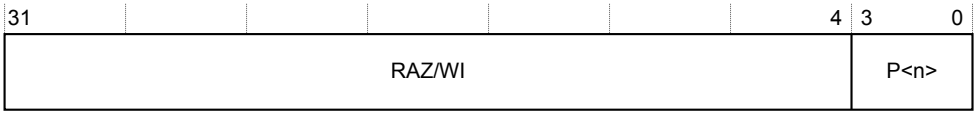

Figure D8-3 AMCNTENCLR0_EL0 bit assignments

RAZ/WI, [31:4]

Read-As-Zero, Writes Ignored

P<n>, [3:0]

AMEVCNTR0n_EL0 disable bit. The possible values are:

0 When this bit is read, the activity counter n is disabled. When it is written, it has no effect.

1 When this bit is read, the activity counter n is enabled. When it is written, it disables the activity counter n.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMCNTENCLR0_EL0

To access AMCNTENCLR0_EL0:

```
MRS <Xt>, AMCNTENCLR0_EL0 ; Read AMCNTENCLR0_EL0 into Xt
```

Register access is encoded as follows:

Table D8-4 AMCNTENCLR0_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c2	4

AMCNTENCLR0_EL0 can be accessed through the external debug interface, offset `0xC20`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.5 AMCNTENCLR1_EL0, Activity Monitors Count Enable Clear Register 1, EL0

AMCNTENCLR1_EL0 disables the control bits for the auxiliary activity monitor counters, AMEVCNTR1<0-2>_EL0.

Bit field descriptions

AMCNTENCLR1_EL0 is a 32-bit register.

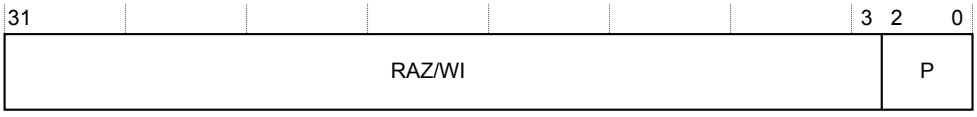

Figure D8-4 AMCNTENCLR1_EL0 bit assignments

RAZ/WI, [31:3]

Read-As-Zero, Writes Ignored

P<n>, [2:0]

AMEVCNTR1<0-2>_EL0 disable bit. The possible values are:

0 When this bit is read, the activity counter n is disabled. When it is written, it has no effect.

1 When this bit is read, the activity counter n is enabled. When it is written, it disables the activity counter n.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMCNTENCLR1_EL0

To access AMCNTENCLR1_EL0:

```
MRS <Xt>, AMCNTENCLR1_EL0 ; Read AMCNTENCLR1_EL0 into Xt
```

Register access is encoded as follows:

Table D8-5 AMCNTENCLR1_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c3	0

AMCNTENCLR1_EL0 can be accessed through the external debug interface, offset `0xC24`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both security states.

`0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.6 AMCNTENSET0_EL0, Activity Monitors Count Enable Set Register 0, EL0

AMCNTENSET0_EL0 enables the activity monitor counters implemented, AMEVCNTR0<0-3>_EL0.

Bit field descriptions

AMCNTENSET0_EL0 is a 32-bit register.

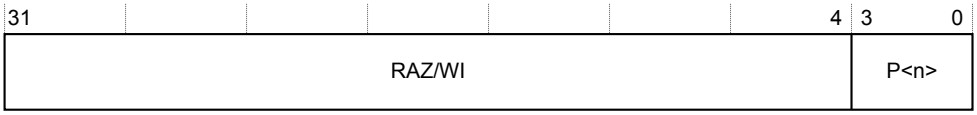

Figure D8-5 AMCNTENSET0_EL0 bit assignments

RAZ/WI, [31:4]

Read-As-Zero, Writes Ignored

P<n>, [3:0]

AMEVCNTR0<0-3>_EL0 enable bit. The possible values are:

0 When this bit is read, the activity counter n is disabled. When it is written, it has no effect.

1 When this bit is read, the activity counter n is enabled. When it is written, it enables the activity counter n.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMCNTENSET0_EL0

To access AMCNTENSET0_EL0:

```
MRS <Xt>, AMCNTENSET0_EL0 ; Read AMCNTENSET0_EL0 into Xt
```

Register access is encoded as follows:

Table D8-6 AMCNTENSET0_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c2	5

AMCNTENSET0_EL0 can be accessed through the external debug interface, offset `0xC00`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.7 AMCNTENSET1_EL0, Activity Monitors Count Enable Set Register 1, EL0

AMCNTENSET1_EL0 enables the control bits for the auxiliary activity monitor counters, AMEVCNTR1<0-2>_EL0.

Bit field descriptions

AMCNTENSET1_EL0 is a 32-bit register.

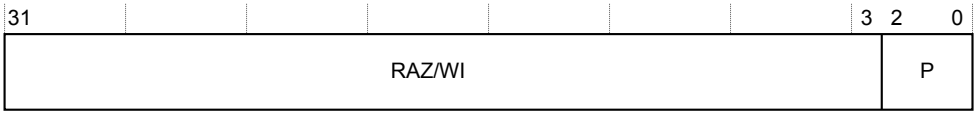

Figure D8-6 AMCNTENSET1_EL0 bit assignments

RAZ/WI, [31:3]

Read-As-Zero, Writes Ignored

P<n>, [2:0]

AMEVCNTR1<0-2>_EL0 enable bit. The possible values are:

0 When this bit is read, the activity counter n is disabled. When it is written, it has no effect.

1 When this bit is read, the activity counter n is enabled. When it is written, it enables the activity counter n.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMCNTENSET1_EL0

To access AMCNTENSET1_EL0:

```
MRS <Xt>, AMCNTENSET1_EL0 ; Read AMCNTENSET1_EL0 into Xt
```

Register access is encoded as follows:

Table D8-7 AMCNTENSET1_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c3	1

AMCNTENSET1_EL0 can be accessed through the external debug interface, offset `0xC04`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both security states.

`0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.8 AMCR_EL0, Activity Monitors Control Register, EL0

AMCR_EL0 is the global control register for the activity monitors.

Bit field descriptions

AMCR_EL0 is a 32-bit register.

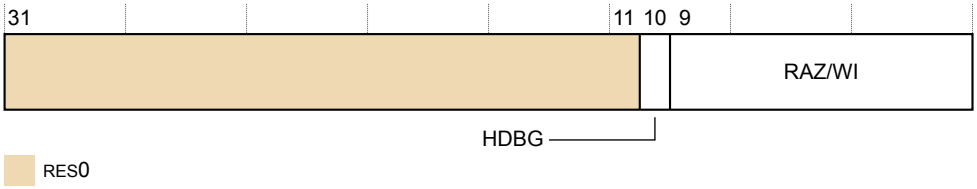

Figure D8-7 AMCR_EL0 bit assignments

RES0, [31:11]

RES0 Reserved

HDBG, [10]

This bit controls whether activity monitor counting is halted when the core is halted in Debug state.

`0b0` Activity Monitors do not stop counting when the core is stopped in Debug state.

`0b1` Activity Monitors stop counting when the core is stopped in Debug state.

RAZ/WI, [9:0]

Read-As-Zero, Writes Ignored

Configurations

There are no configuration notes.

Usage constraints

Accessing AMCR_EL0

To access AMCR_EL0:

```
MRS <Xt>, AMCR_EL0 ; Read AMCR_EL0 into Xt
```

Register access is encoded as follows:

Table D8-8 AMCR_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c2	0

AMCR_EL0 can be accessed through the external debug interface, offset `0xE04`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both Security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

  `0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps Non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.9 AMEVCNTR0n_EL0, Activity Monitors Event Counter Registers 0n, EL0

AMEVCNTR0n_EL0 registers provide access to the architected, required activity monitor event counters.

Bit field descriptions

AMEVCNTR0n_EL0 are 64-bit registers.

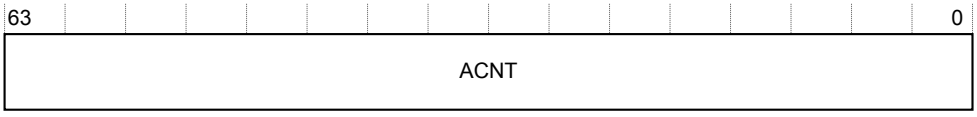

Figure D8-8 AMEVCNTR0n_EL0 bit assignments

ACNT, [63:0]

Value of the activity counter AMEVCNTR0n_EL0.

This bit field resets to zero and the counters monitoring cycle events do not increment when the core is in *Wait For Interrupt* (WFI) or *Wait For Event* (WFE).

Configurations

There are no configuration notes.

Usage constraints

Accessing AMEVCNTR0n_EL0

To access AMEVCNTR0n_EL0:

```
MRS <Xt>, AMEVCNTR0n_EL0 ; Read AMEVCNTR0n_EL0 into Xt
```

Register access is encoded as follows:

Table D8-9 AMEVCNTR0n_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c4	<n>

AMEVCNTR0n_EL0[63:32] can also be accessed through the external memory-mapped interface, offset `0x004+8n`. In this case, it is read-only.

AMEVCNTR0n_EL0[31:0] can also be accessed through the external memory-mapped interface, offset `0x000+8n`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both Security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps Non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.10 AMEVCNTR1n_EL0, Activity Monitors Event Counter Registers 1n, EL0

AMEVCNTR1n_EL0 provides access to the auxiliary activity monitor event counters. The Cortex-A78C core implements three auxiliary counters so n is 0-2.

Bit field descriptions

AMEVCNTR1n_EL0 is a 64-bit register.

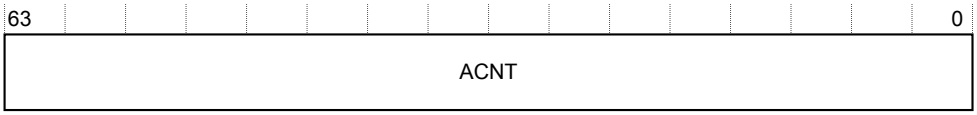

Figure D8-9 AMEVCNTR1n_EL0 bit assignments

ACNT, [63:0]

Value of the activity counter AMEVCNTR1n_EL0.

This bit field resets to zero and the counters monitoring cycle events do not increment when the core is in WFI or WFE.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMEVCNTR1n_EL0

To access AMEVCNTR1n_EL0:

```
MRS <Xt>, AMEVCNTR1n_EL0 ; Read AMEVCNTR1n_EL0 into Xt
```

Register access is encoded as follows:

Table D8-10 AMEVCNTR1n_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c12	n

AMEVCNTR1n_EL0[63:32] can also be accessed through the external memory-mapped interface, offset `0x104+8n`. In this case, it is read-only.

AMEVCNTR1n_EL0[31:0] can also be accessed through the external memory-mapped interface, offset `0x100+8n`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.11 AMEVTYPER0n_EL0, Activity Monitors Event Type Registers 0n, EL0

AMEVTYPER0n_EL0 registers provide information on the events that an architected, required activity monitor counter AMEVCNTR0n_EL0 counts.

Bit field descriptions

AMEVTYPER0n_EL0 are 32-bit registers.

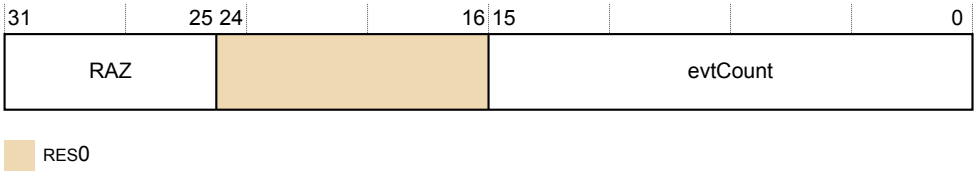

Figure D8-10 AMEVTYPER0n_EL0 bit assignments

RAZ/WI, [31:25]

Read-As-Zero, Writes Ignored

RES0, [24:16]

RES0 Reserved

evtCount, [15:0]

The event the counter monitors might be fixed at implementation. In this case, the field is read-only. See *C3.4 AMU events* on page C3-443.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMEVTYPER0n_EL0

To access AMEVTYPER0n_EL0:

```
MRS <Xt>, AMEVTYPER0n_EL0 ; Read AMEVTYPER0n_EL0 into Xt
```

Register access is encoded as follows:

Table D8-11 AMEVTYPER0n_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c6	<n>

AMEVTYPER0n_EL0 can be accessed through the external debug interface, offset `0x400+4n`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both Security states.

`0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps Non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.12 AMEVTYPER1n_EL0, Activity Monitors Event Type Registers 1n, EL0

AMEVTYPER1n_EL0 configures the event an auxiliary activity counter monitors, or defines fixed-at-implementation event selection. The Cortex-A78C core implements three auxiliary counters so n is 0-2.

Bit field descriptions

AMEVTYPER1n_EL0 is a 32-bit register.

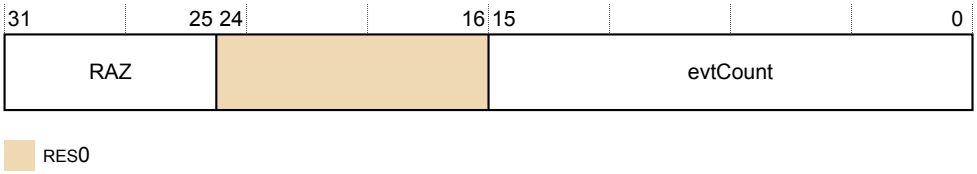

Figure D8-11 AMEVTYPER1n_EL0 bit assignments

RAZ/WI, [31:25]

Read-As-Zero, Writes Ignored

RES0, [24:16]

RES0 Reserved

evtCount, [15:0]

The event the counter monitors might be fixed at implementation. In this case, the field is read-only. See *C3.4 AMU events* on page C3-443.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMEVTYPER1n_EL0

To access AMEVTYPER1n_EL0:

```
MRS <Xt>, AMEVTYPER1n_EL0 ; Read AMEVTYPER1n_EL0 into Xt
```

Register access is encoded as follows:

Table D8-12 AMEVTYPER1n_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c14	n

AMEVTYPER1n_EL0 can be accessed through the external debug interface, offset `0x480+4n`. In this case, it is read-only.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both Security states.

`0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

`0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps Non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

D8.13 AMUSERENR_EL0, Activity Monitors User Enable Register, EL0

AMUSERENR_EL0 is the global user enable register for the activity monitors. It enables or disables EL0 access to the activity monitors. AMUSERENR_EL0 is applicable to both the architected and the auxiliary event counter groups.

Bit field descriptions

AMUSERENR_EL0 is a 32-bit register.

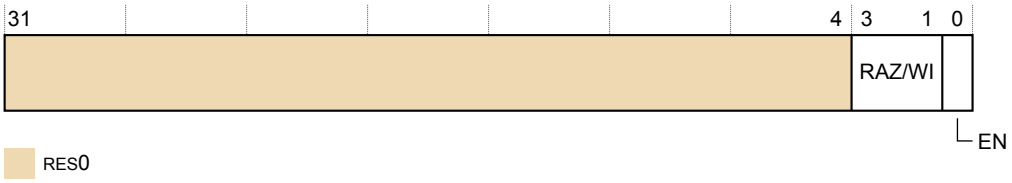

Figure D8-12 AMUSERENR_EL0 bit assignments

RES0, [31:4]

RES0 Reserved

RAZ/WI, [3:1]

Read-As-Zero, Writes Ignored

EN, [0]

Traps EL0 accesses to the activity monitor registers to EL1. The possible values are:

0 EL0 accesses to the activity monitor registers are trapped to EL1.

1 EL0 accesses to the activity monitor registers are not trapped to EL1. Software can access all activity monitor registers at EL0.

Configurations

There are no configuration notes.

Usage constraints

Accessing AMUSERENR_EL0

To access AMUSERENR_EL0:

```
MRS <Xt>, AMUSERENR_EL0 ; Read AMUSERENR_EL0 into Xt
```

Register access is encoded as follows:

Table D8-13 AMUSERENR_EL0 encoding

op0	op1	CRn	CRm	op2
3	3	c15	c2	3

AMUSERENR_EL0 is excluded from the memory mapped view.

This register is accessible as follows:

EL0	EL1	EL2	EL3
RO	RO	RO	RO

Traps and enables

The following control bits for traps on accesses to activity monitor registers are introduced:

- The ACTLR_EL3.TAM (trap activity monitor access) bit traps EL2, EL1 and EL0 accesses to all activity monitor registers to EL3, from both Security states.

  `0b0` EL2, EL1 and EL0 accesses to all activity monitor registers are not trapped to EL3.

  `0b1` EL2, EL1 and EL0 accesses to all activity monitor registers are trapped to EL3.

- The ACTLR_EL2.TAM (trap activity monitor access) bit traps Non-secure EL1 and EL0 accesses to all activity monitor registers to EL2.

  `0b0` Non-secure accesses from EL1 and EL0 to activity monitor registers are not trapped to EL2.

  `0b1` Non-secure accesses from EL1 and EL0 to activity monitor registers are trapped to EL2.

The TAM bit is allocated to bit [30] of ACTLR_EL3 and ACTLR_EL2.

The previously defined ACTLR_ELx.AMEN bits are now RES0.

Chapter D9
Memory-mapped AMU registers

This chapter describes the memory-mapped *Activity Monitor Unit* (AMU) registers. The memory-mapped interface provides read-only access to the AMU registers via the external debug interface.

It contains the following sections:

D9.1 Memory-mapped AMU register summary

There are *Activity Monitor Unit* (AMU) registers that are accessible through the external debug interface.

These registers are listed in the following table. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Table D9-1 Memory-mapped AMU register summary

Offset	Name	Type	Description
[63:32]: `0x004+8n` [31:0]: `0x000+8n`	AMEVCNTR0n	RO	*D8.9 AMEVCNTR0n_EL0, Activity Monitors Event Counter Registers 0n, EL0* on page D8-566
[63:32]: `0x104+8n` [31:0]: `0x100+8n`	AMEVCNTR1n	RO	*D8.10 AMEVCNTR1n_EL0, Activity Monitors Event Counter Registers 1n, EL0* on page D8-568
`0x400+4n`	AMEVTYPER0n	RO	*D8.11 AMEVTYPER0n_EL0, Activity Monitors Event Type Registers 0n, EL0* on page D8-570
`0x480+4n`	AMEVTYPER1n	RO	*D8.12 AMEVTYPER1n_EL0, Activity Monitors Event Type Registers 1n, EL0* on page D8-572
`0xC00`	AMCNTENSET0	RO	*D8.6 AMCNTENSET0_EL0, Activity Monitors Count Enable Set Register 0, EL0* on page D8-560
`0xC04`	AMCNTENSET1	RO	*D8.7 AMCNTENSET1_EL0, Activity Monitors Count Enable Set Register 1, EL0* on page D8-562
`0xC20`	AMCNTENCLR0	RO	*D8.4 AMCNTENCLR0_EL0, Activity Monitors Count Enable Clear Register 0, EL0* on page D8-556
`0xC24`	AMCNTENCLR1	RO	*D8.5 AMCNTENCLR1_EL0, Activity Monitors Count Enable Clear Register 1, EL0* on page D8-558
`0xCE0`	AMCGCR	RO	*D8.3 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register, EL0* on page D8-554
`0xE00`	AMCFGR	RO	*D8.2 AMCFGR_EL0, Activity Monitors Configuration Register, EL0* on page D8-552
`0xE04`	AMCR	RO	*D8.8 AMCR_EL0, Activity Monitors Control Register, EL0* on page D8-564
`0xE08`	AMIIDR	RO	*D9.10 AMIIDR, Activity Monitors Implementation Identification Register* on page D9-588
`0xFA8`	AMDEVAFF0	RO	*D9.6 AMDEVAFF0, Activity Monitors Device Affinity Register 0* on page D9-584

Table D9-1 Memory-mapped AMU register summary (continued)

Offset	Name	Type	Description
0xFAC	AMDEVAFF1	RO	*D9.7 AMDEVAFF1, Activity Monitors Device Affinity Register 1* on page D9-585
0xFBC	AMDEVARCH	RO	*D9.8 AMDEVARCH, Activity Monitors Device Architecture Register* on page D9-586
0xFCC	AMDEVTYPE	RO	*D9.9 AMDEVTYPE, Activity Monitors Device Type Register* on page D9-587
0xFD0	AMPIDR4	RO	*D9.15 AMPIDR4, Activity Monitors Peripheral Identification Register 4* on page D9-593
0xFE0	AMPIDR0	RO	*D9.11 AMPIDR0, Activity Monitors Peripheral Identification Register 0* on page D9-589
0xFE4	AMPIDR1	RO	*D9.12 AMPIDR1, Activity Monitors Peripheral Identification Register 1* on page D9-590
0xFE8	AMPIDR2	RO	*D9.13 AMPIDR2, Activity Monitors Peripheral Identification Register 2* on page D9-591
0xFEC	AMPIDR3	RO	*D9.14 AMPIDR3, Activity Monitors Peripheral Identification Register 3* on page D9-592
0xFF0	AMCIDR0	RO	*D9.2 AMCIDR0, Activity Monitors Component Identification Register 0* on page D9-580
0xFF4	AMCIDR1	RO	*D9.3 AMCIDR1, Activity Monitors Component Identification Register 1* on page D9-581
0xFF8	AMCIDR2	RO	*D9.4 AMCIDR2, Activity Monitors Component Identification Register 2* on page D9-582
0xFFC	AMCIDR3	RO	*D9.5 AMCIDR3, Activity Monitors Component Identification Register 3* on page D9-583

D9.2 AMCIDR0, Activity Monitors Component Identification Register 0

The AMCIDR0 provides information to identify an activities monitors component.

Bit field descriptions

The AMCIDR0 is a 32-bit register.

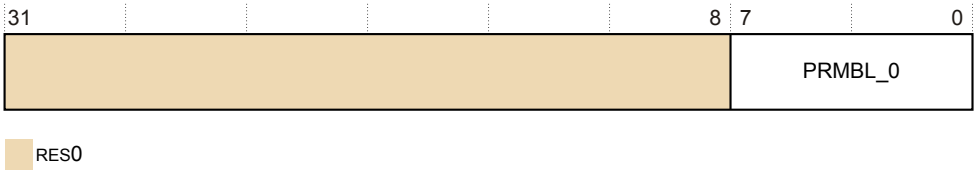

Figure D9-1 AMCIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_0, [7:0]

`0x0D` Preamble byte 0

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

D9.3 AMCIDR1, Activity Monitors Component Identification Register 1

The AMCIDR1 provides information to identify an activity monitors component.

Bit field descriptions

The AMCIDR1 is a 32-bit register.

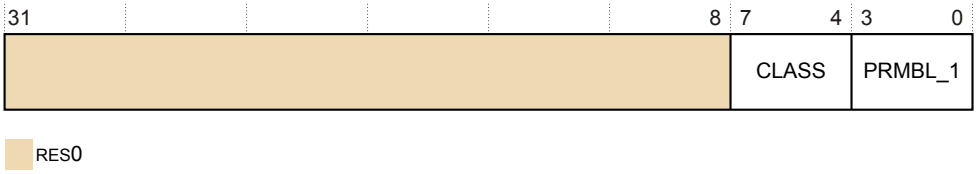

Figure D9-2 AMCIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

CLASS, [7:4]

`0x9` CoreSight component

PRMBL_1, [3:0]

`0x0` Preamble

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

D9.4 AMCIDR2, Activity Monitors Component Identification Register 2

The AMCIDR2 provides information to identify an activity monitors component.

Bit field descriptions

The AMCIDR2 is a 32-bit register.

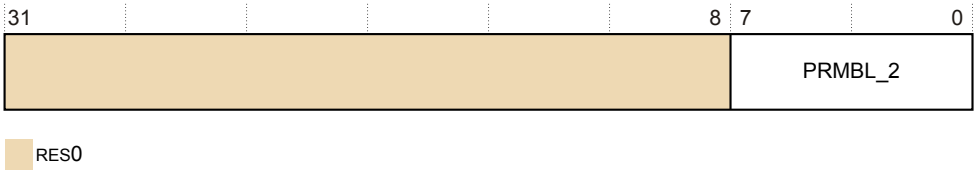

Figure D9-3 AMCIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_2, [7:0]

`0x05` Preamble byte 2

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

D9.5 AMCIDR3, Activity Monitors Component Identification Register 3

The AMCIDR3 provides information to identify an activity monitors component.

Bit field descriptions

The AMCIDR3 is a 32-bit register.

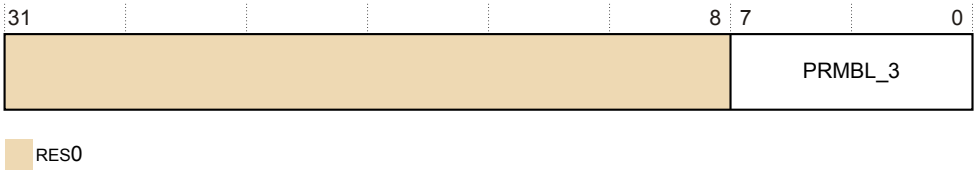

Figure D9-4 EDCIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_3, [7:0]

`0xB1` Preamble byte 3

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

D9.6 AMDEVAFF0, Activity Monitors Device Affinity Register 0

The AMDEVAFF0 provides an additional core identification mechanism for scheduling purposes in a cluster. AMDEVAFF0 is a read-only copy of MPIDR_EL1[31:0] accessible from the external debug interface.

Bit field descriptions

The AMDEVAFF0 is a 32-bit register and is a copy of the MPIDR register. See *B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314 for full bit field descriptions.

D9.7 AMDEVAFF1, Activity Monitors Device Affinity Register 1

The AMDEVAFF1 provides an additional core identification mechanism for scheduling purposes in a cluster. AMDEVAFF1 is a read-only copy of MPIDR_EL1[63:32] accessible from the external debug interface.

Bit field descriptions

The AMDEVAFF1 is a 32-bit register and is a copy of the MPIDR register. See *B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314 for full bit field descriptions.

D9.8 AMDEVARCH, Activity Monitors Device Architecture Register

The AMDEVARCH identifies the programmers' model architecture of the *Activity Monitor Unit* (AMU) component.

Bit field descriptions

The AMDEVARCH is a 32-bit register.

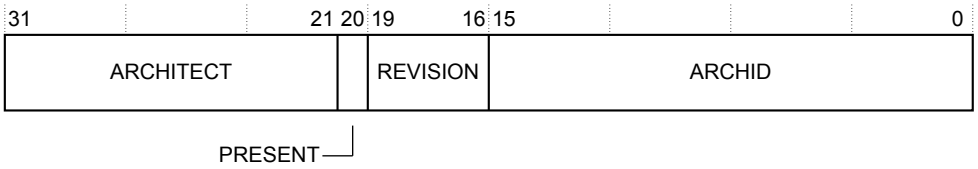

Figure D9-5 AMDEVARCH bit assignments

ARCHITECT, [31:21]

Defines the architect of the component:

[31:28] `0x4`, Arm JEP continuation

[27:21] `0x3B`, Arm JEP 106 code

PRESENT, [20]

Indicates the presence of this register:

`0b1` Register is present.

REVISION, [19:16]

Architecture revision:

`0x00` Architecture revision 1

ARCHID, [15:0]

Architecture ID:

`0x0A66` AMU architecture version AMUv1

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMDEVARCH can be accessed through the external debug interface, offset `0xFBC`.

D9.9 AMDEVTYPE, Activity Monitors Device Type Register

The AMDEVTYPE indicates to a debugger that this component is part of a core's performance monitor interface.

Bit field descriptions

The AMDEVTYPE is a 32-bit register.

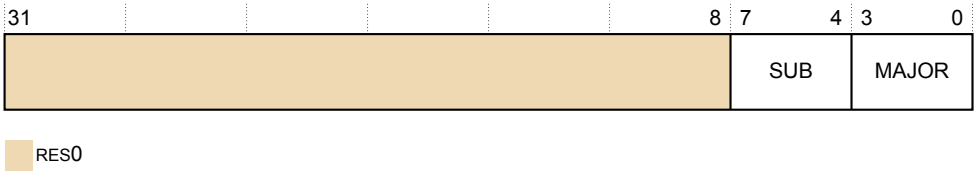

Figure D9-6 AMDEVTYPE bit assignments

RES0, [31:8]

RES0 Reserved

SUB, [7:4]

The sub-type of the component:

`0x1` Indicates this is a component within a core.

MAJOR, [3:0]

The main type of the component:

`0x6` Indicates this is a performance monitor component.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMDEVTYPE can be accessed through the external debug interface, offset `0xFCC`.

D9.10 AMIIDR, Activity Monitors Implementation Identification Register

The AMIIDR defines the implementer and revisions of the *Activity Monitor Unit* (AMU).

Bit field descriptions

AMIIDR is a 32-bit register.

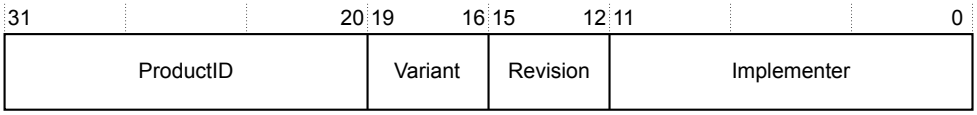

Figure D9-7 AMIIDR_EL1 bit assignments

ProductID, [31:20]

Indicates the AMU part identifier. This value is:

`0xD4B` The AMU part number

Variant, [19:16]

Indicates the variant number of the core. This is the major revision number *x* in the r*x* part of the r*x*p*y* description of the product revision status. This value is:

`0x0` r0p1

Revision, [15:12]

Indicates the minor revision number of the core. This is the minor revision number *y* in the p*y* part of the r*x*p*y* description of the product revision status. This value is:

`0x1` r0p1

Implementer, [11:0]

Indicates the JEP106 code of the company that implemented the AMU. This value is:

`0x43B` Bits [11:8] contain the JEP106 continuation code of the implementer.

Bit [7] is RES0.

Bits [6:0] contain the JEP106 identity code of the implementer.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMIIDR can be accessed through the external debug interface, offset `0xE08`.

D9.11 AMPIDR0, Activity Monitors Peripheral Identification Register 0

The AMPIDR0 provides information to identify an activity monitors component.

Bit field descriptions

The AMPIDR0 is a 32-bit register.

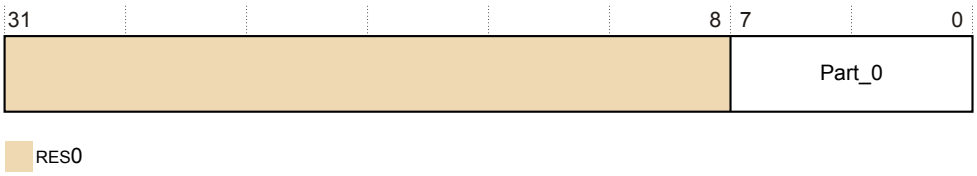

Figure D9-8 AMPIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

Part_0, [7:0]

`0x4B` Least significant byte of the *Activity Monitor Unit* (AMU) part number.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

D9.12 AMPIDR1, Activity Monitors Peripheral Identification Register 1

The AMPIDR1 provides information to identify an activity monitors component.

Bit field descriptions

The AMPIDR1 is a 32-bit register.

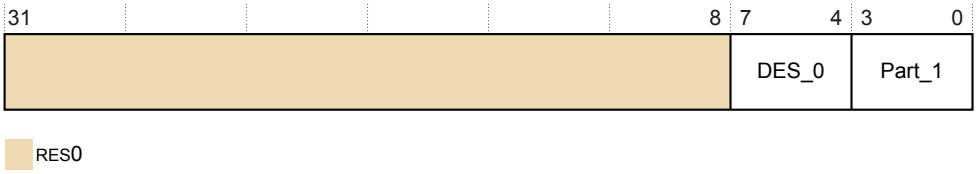

Figure D9-9 AMPIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

DES_0, [7:4]

`0xB` Arm Limited. This is bits[3:0] of JEP106 ID code.

Part_1, [3:0]

`0xD` Most significant four bits of the *Activity Monitor Unit* (AMU) part number

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

D9.13 AMPIDR2, Activity Monitors Peripheral Identification Register 2

The AMPIDR2 provides information to identify an activity monitors component.

Bit field descriptions

The AMPIDR2 is a 32-bit register.

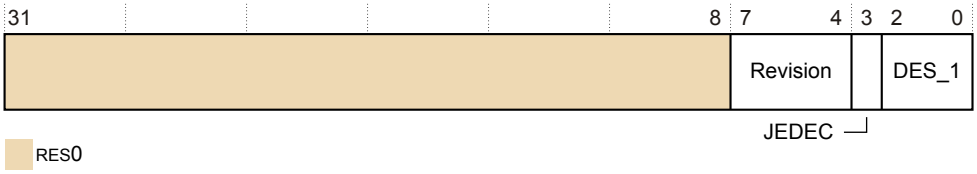

Figure D9-10 AMPIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

Revision, [7:4]

`0x1` r0p1

JEDEC, [3]

`0b1` RES1. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is bits[6:4] of JEP106 ID code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

D9.14 AMPIDR3, Activity Monitors Peripheral Identification Register 3

The AMPIDR3 provides information to identify an activity monitors component.

Bit field descriptions

The AMPIDR3 is a 32-bit register.

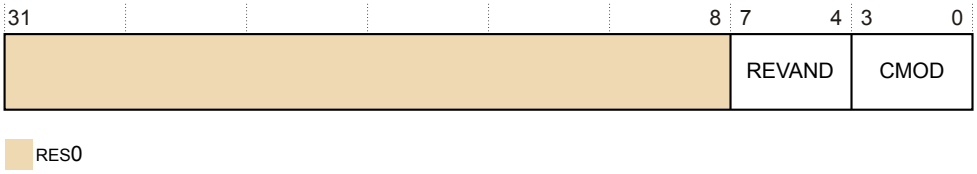

Figure D9-11 AMPIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

REVAND, [7:4]

`0x0` Part minor revision

CMOD, [3:0]

`0x0` Not customer modified

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

D9.15 AMPIDR4, Activity Monitors Peripheral Identification Register 4

The AMPIDR4 provides information to identify an activity monitors component.

Bit field descriptions

The AMPIDR4 is a 32-bit register.

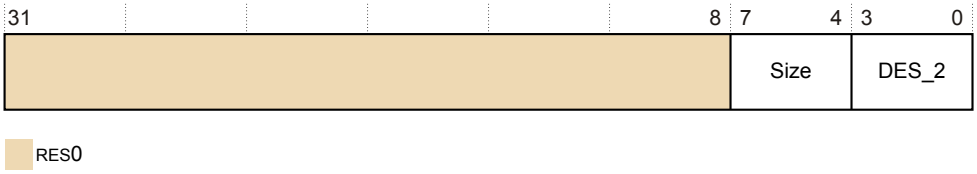

Figure D9-12 AMPIDR4 bit assignments

RES0, [31:8]

RES0 Reserved

Size, [7:4]

`0x0` Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited. This is bits[3:0] of the JEP106 continuation code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The AMPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

Chapter D10
ETM registers

This chapter describes the *Embedded Trace Macrocell* (ETM) registers.

It contains the following sections:

D10.1 ETM register summary

This section summarizes the *Embedded Trace Macrocell* (ETM) trace unit registers.

All ETM trace unit registers are 32-bit wide. The description of each register includes its offset from a base address. The base address is defined by the system integrator when placing the ETM trace unit in the Debug-APB memory map.

The following table lists all of the ETM trace unit registers.

Table D10-1 ETM trace unit register summary

Offset	Name	Type	Reset	Description
0x004	TRCPRGCTLR	RW	0x00000000	*D10.61 TRCPRGCTLR, Programming Control Register* on page D10-672
0x00C	TRCSTATR	RO	0x00000003	*D10.68 TRCSTATR, Status Register* on page D10-680
0x010	TRCCONFIGR	RW	UNK	*D10.20 TRCCONFIGR, Trace Configuration Register* on page D10-623
0x018	TRCAUXCTLR	RW	0x00000000	*D10.5 TRCAUXCTLR, Auxiliary Control Register* on page D10-605
0x020	TRCEVENTCTL0R	RW	UNK	*D10.26 TRCEVENTCTL0R, Event Control 0 Register* on page D10-631
0x024	TRCEVENTCTL1R	RW	UNK	*D10.27 TRCEVENTCTL1R, Event Control 1 Register* on page D10-633
0x030	TRCTSCTLR	RW	UNK	*D10.71 TRCTSCTLR, Global Timestamp Control Register* on page D10-683
0x034	TRCSYNCPR	RW	UNK	*D10.69 TRCSYNCPR, Synchronization Period Register* on page D10-681
0x038	TRCCCCTLR	RW	UNK	*D10.7 TRCCCCTLR, Cycle Count Control Register* on page D10-608
0x03C	TRCBBCTLR	RW	UNK	*D10.6 TRCBBCTLR, Branch Broadcast Control Register* on page D10-607
0x040	TRCTRACEIDR	RW	UNK	*D10.70 TRCTRACEIDR, Trace ID Register* on page D10-682
0x080	TRCVICTLR	RW	UNK	*D10.72 TRCVICTLR, ViewInst Main Control Register* on page D10-684
0x084	TRCVIIECTLR	RW	UNK	*D10.73 TRCVIIECTLR, ViewInst Include-Exclude Control Register* on page D10-686
0x088	TRCVISSCTLR	RW	UNK	*D10.74 TRCVISSCTLR, ViewInst Start-Stop Control Register* on page D10-687
0x100	TRCSEQEVR0	RW	UNK	*D10.63 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2* on page D10-674
0x104	TRCSEQEVR1	RW	UNK	*D10.63 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2* on page D10-674
0x108	TRCSEQEVR2	RW	UNK	*D10.63 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2* on page D10-674
0x118	TRCSEQRSTEVR	RW	UNK	*D10.64 TRCSEQRSTEVR, Sequencer Reset Control Register* on page D10-676
0x11C	TRCSEQSTR	RW	UNK	*D10.65 TRCSEQSTR, Sequencer State Register* on page D10-677
0x120	TRCEXTINSELR	RW	UNK	*D10.28 TRCEXTINSELR, External Input Select Register* on page D10-634

Table D10-1 ETM trace unit register summary (continued)

Offset	Name	Type	Reset	Description
0x140	TRCCNTRLDVR0	RW	UNK	*D10.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1* on page D10-621
0x144	TRCCNTRLDVR1	RW	UNK	*D10.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1* on page D10-621
0x150	TRCCNTCTLR0	RW	UNK	*D10.16 TRCCNTCTLR0, Counter Control Register 0* on page D10-617
0x154	TRCCNTCTLR1	RW	UNK	*D10.17 TRCCNTCTLR1, Counter Control Register 1* on page D10-619
0x160	TRCCNTVR0	RW	UNK	*D10.19 TRCCNTVRn, Counter Value Registers 0-1* on page D10-622
0x164	TRCCNTVR1	RW	UNK	*D10.19 TRCCNTVRn, Counter Value Registers 0-1* on page D10-622
0x180	TRCIDR8	RO	0x00000000	*D10.35 TRCIDR8, ID Register 8* on page D10-646
0x184	TRCIDR9	RO	0x00000000	*D10.36 TRCIDR9, ID Register 9* on page D10-647
0x188	TRCIDR10	RO	0x00000000	*D10.37 TRCIDR10, ID Register 10* on page D10-648
0x18C	TRCIDR11	RO	0x00000000	*D10.38 TRCIDR11, ID Register 11* on page D10-649
0x190	TRCIDR12	RO	0x00000000	*D10.39 TRCIDR12, ID Register 12* on page D10-650
0x194	TRCIDR13	RO	0x00000000	*D10.40 TRCIDR13, ID Register 13* on page D10-651
0x1C0	TRCIMSPEC0	RW	0x00000000	*D10.41 TRCIMSPEC0, IMPLEMENTATION SPECIFIC Register 0* on page D10-652
0x1E0	TRCIDR0	RO	0x28000EA1	*D10.29 TRCIDR0, ID Register 0* on page D10-635
0x1E4	TRCIDR1	RO	0x4100F423	*D10.30 TRCIDR1, ID Register 1* on page D10-637
0x1E8	TRCIDR2	RO	0x20001088	*D10.31 TRCIDR2, ID Register 2* on page D10-638
0x1EC	TRCIDR3	RO	0x017B0004	*D10.32 TRCIDR3, ID Register 3* on page D10-640
0x1F0	TRCIDR4	RO	0x11170004	*D10.33 TRCIDR4, ID Register 4* on page D10-642
0x1F4	TRCIDR5	RO	0x284708D6	*D10.34 TRCIDR5, ID Register 5* on page D10-644
0x200	TRCRSCTLRn	RW	UNK	*D10.62 TRCRSCTLRn, Resource Selection Control Registers 2-16* on page D10-673, n is 2, 15
0x280	TRCSSCCR0	RW	UNK	*D10.66 TRCSSCCR0, Single-Shot Comparator Control Register 0* on page D10-678
0x2A0	TRCSSCSR0	RW	UNK	*D10.67 TRCSSCSR0, Single-Shot Comparator Status Register 0* on page D10-679
0x300	TRCOSLAR	WO	0x00000001	*D10.51 TRCOSLAR, OS Lock Access Register* on page D10-662
0x304	TRCOSLSR	RO	0x0000000A	*D10.52 TRCOSLSR, OS Lock Status Register* on page D10-663
0x310	TRCPDCR	RW	0x00000000	*D10.53 TRCPDCR, Power Down Control Register* on page D10-664
0x314	TRCPDSR	RO	0x00000023	*D10.54 TRCPDSR, Power Down Status Register* on page D10-665
0x400	TRCACVRn	RW	UNK	*D10.3 TRCACVRn, Address Comparator Value Registers 0-7* on page D10-603

Table D10-1 ETM trace unit register summary (continued)

Offset	Name	Type	Reset	Description
0x480	TRCACATRn	RW	UNK	*D10.2 TRCACATRn, Address Comparator Access Type Registers 0-7* on page D10-601
0x600	TRCCIDCVR0	RW	UNK	*D10.9 TRCCIDCVR0, Context ID Comparator Value Register 0* on page D10-610
0x640	TRCVMIDCVR0	RW	UNK	*D10.75 TRCVMIDCVR0, VMID Comparator Value Register 0* on page D10-688
0x680	TRCCIDCCTLR0	RW	UNK	*D10.8 TRCCIDCCTLR0, Context ID Comparator Control Register 0* on page D10-609
0x688	TRCVMIDCCTRL0	RW	UNK	*D10.76 TRCVMIDCCTLR0, Virtual context identifier Comparator Control Register 0* on page D10-689
0xEDC	TRCITMISCOUT	WO	UNK	*D10.48 TRCITMISCOUT, Trace Integration Miscellaneous Outputs Register* on page D10-659
0xEE0	TRCITMISCIN	RO	UNK	*D10.47 TRCITMISCIN, Trace Integration Miscellaneous Input Register* on page D10-658
0xEEC	TRCITATBDATA0	WO	UNK	*D10.45 TRCITATBDATA0, Trace Integration Test ATB Data Register 0* on page D10-656
0xEF0	TRCITATBCTR2	RO	UNK	*D10.44 TRCITATBCTR2, Trace Integration Test ATB Control Register 2* on page D10-655
0xEF4	TRCITATBCTR1	WO	UNK	*D10.43 TRCITATBCTR1, Trace Integration Test ATB Control Register 1* on page D10-654
0xEF8	TRCITATBCTR0	WO	UNK	*D10.42 TRCITATBCTR0, Trace Integration Test ATB Control Register 0* on page D10-653
0xF00	TRCITCTRL	RW	0x00000000	*D10.46 TRCITCTRL, Trace Integration Mode Control register* on page D10-657
0xFA0	TRCCLAIMSET	RW	UNK	*D10.15 TRCCLAIMSET, Claim Tag Set Register* on page D10-616
0xFA4	TRCCLAIMCLR	RW	0x00000000	*D10.14 TRCCLAIMCLR, Claim Tag Clear Register* on page D10-615
0xFA8	TRCDEVAFF0	RO	UNK	*D10.21 TRCDEVAFF0, Device Affinity Register 0* on page D10-626
0xFAC	TRCDEVAFF1	RO	UNK	*D10.22 TRCDEVAFF1, Device Affinity Register 1* on page D10-627
0xFB0	TRCLAR	WO	UNK	*D10.49 TRCLAR, Software Lock Access Register* on page D10-660
0xFB4	TRCLSR	RO	0x00000000	*D10.50 TRCLSR, Software Lock Status Register* on page D10-661
0xFB8	TRCAUTHSTATUS	RO	UNK	*D10.4 TRCAUTHSTATUS, Authentication Status Register* on page D10-604
0xFBC	TRCDEVARCH	RO	0x47724A13	*D10.23 TRCDEVARCH, Device Architecture Register* on page D10-628
0xFC8	TRCDEVID	RO	0x00000000	*D10.24 TRCDEVID, Device ID Register* on page D10-629
0xFCC	TRCDEVTYPE	RO	0x00000013	*D10.25 TRCDEVTYPE, Device Type Register* on page D10-630
0xFE0	TRCPIDR0	RO	0x0000000B	*D10.55 TRCPIDR0, ETM Peripheral Identification Register 0* on page D10-666
0xFE4	TRCPIDR1	RO	0x000000BD	*D10.56 TRCPIDR1, ETM Peripheral Identification Register 1* on page D10-667

Table D10-1 ETM trace unit register summary (continued)

Offset	Name	Type	Reset	Description
0xFE8	TRCPIDR2	RO	0x0000000B	*D10.57 TRCPIDR2, ETM Peripheral Identification Register 2* on page D10-668
0xFEC	TRCPIDR3	RO	0x00000000	*D10.58 TRCPIDR3, ETM Peripheral Identification Register 3* on page D10-669
0xFD0	TRCPIDR4	RO	0x00000004	*D10.59 TRCPIDR4, ETM Peripheral Identification Register 4* on page D10-670
0xFD4-0xFDC	TRCPIDRn	RO	0x00000000	*D10.60 TRCPIDRn, ETM Peripheral Identification Registers 5-7* on page D10-671
0xFF0	TRCCIDR0	RO	0x0000000D	*D10.10 TRCCIDR0, ETM Component Identification Register 0* on page D10-611
0xFF4	TRCCIDR1	RO	0x00000090	*D10.11 TRCCIDR1, ETM Component Identification Register 1* on page D10-612
0xFF8	TRCCIDR2	RO	0x00000005	*D10.12 TRCCIDR2, ETM Component Identification Register 2* on page D10-613
0xFFC	TRCCIDR3	RO	0x000000B1	*D10.13 TRCCIDR3, ETM Component Identification Register 3* on page D10-614

D10.2 TRCACATRn, Address Comparator Access Type Registers 0-7

The TRCACATRn control the access for the corresponding address comparators.

Bit field descriptions

The TRCACATRn is a 64-bit register.

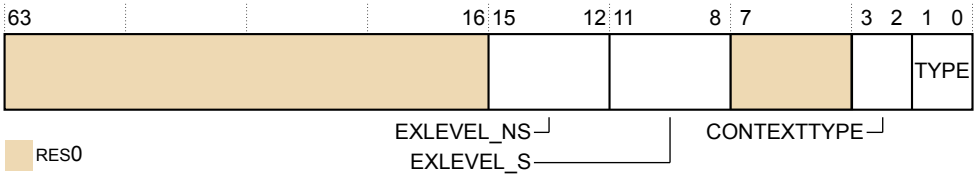

Figure D10-1 TRCACATRn bit assignments

RES0, [63:16]

RES0 Reserved

EXLEVEL_NS, [15:12]

Each bit controls whether a comparison can occur in Non-secure state for the corresponding Exception level. The possible values are:

`0` The trace unit can perform a comparison, in Non-secure state, for Exception level *n*.

`1` The trace unit does not perform a comparison, in Non-secure state, for Exception level *n*.

The Exception levels are:

Bit[12] Exception level 0

Bit[13] Exception level 1

Bit[14] Exception level 2

Bit[15] Always RES0

EXLEVEL_S, [11:8]

Each bit controls whether a comparison can occur in Secure state for the corresponding Exception level. The possible values are:

`0` The trace unit can perform a comparison, in Secure state, for Exception level *n*.

`1` The trace unit does not perform a comparison, in Secure state, for Exception level *n*.

The Exception levels are:

Bit[8] Exception level 0

Bit[9] Exception level 1

Bit[10] Always RES0

Bit[11] Exception level 3

RES0, [7:4]

RES0 Reserved

CONTEXT TYPE, [3:2]

Controls whether the trace unit performs a Context ID comparison, a VMID comparison, or both comparisons:

`0b00` The trace unit does not perform a Context ID comparison.

`0b01` The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.

`0b10` The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.

`0b11` The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

TYPE, [1:0]

Type of comparison:

`0b00` Instruction address, RES0

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCACATR*n* can be accessed through the external debug interface, offset `0x480-0x4B8`.

D10.3 TRCACVRn, Address Comparator Value Registers 0-7

The TRCACVRn indicate the address for the address comparators.

Bit field descriptions

The TRCACVRn is a 64-bit register.

Figure D10-2 TRCACVRn bit assignments

ADDRESS, [63:0]
The address value to compare against

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCACVRn can be accessed through the external debug interface, offset `0x400-0x43C`.

D10.4 TRCAUTHSTATUS, Authentication Status Register

The TRCAUTHSTATUS indicates the current level of tracing permitted by the system.

Bit field descriptions

The TRCAUTHSTATUS is a 64-bit register.

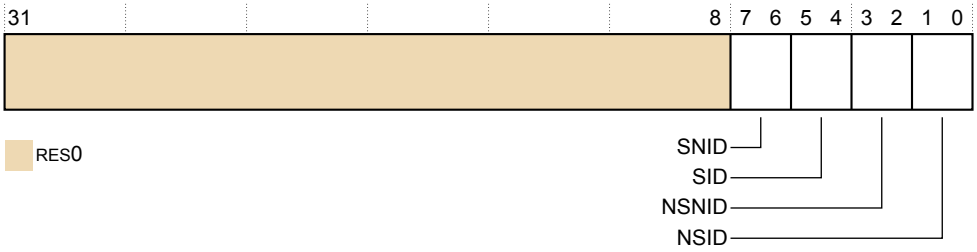

Figure D10-3 TRCAUTHSTATUS bit assignments

RES0, [31:8]

RES0 Reserved

SNID, [7:6]

Secure Non-invasive Debug:

`0b10` Secure Non-invasive Debug implemented but disabled

`0b11` Secure Non-invasive Debug implemented and enabled

SID, [5:4]

Secure Invasive Debug:

`0b00` Secure Invasive Debug is not implemented.

NSNID, [3:2]

Non-secure Non-invasive Debug:

`0b10` Non-secure Non-invasive Debug implemented but disabled, **NIDEN**=0

`0b11` Non-secure Non-invasive Debug implemented and enabled, **NIDEN**=1

NSID, [1:0]

Non-secure Invasive Debug:

`0b00` Non-secure Invasive Debug is not implemented.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCAUTHSTATUS can be accessed through the external debug interface, offset `0xFB8`.

D10.5 TRCAUXCTLR, Auxiliary Control Register

The TRCAUXCTLR provides IMPLEMENTATION DEFINED configuration and control options.

Bit field descriptions

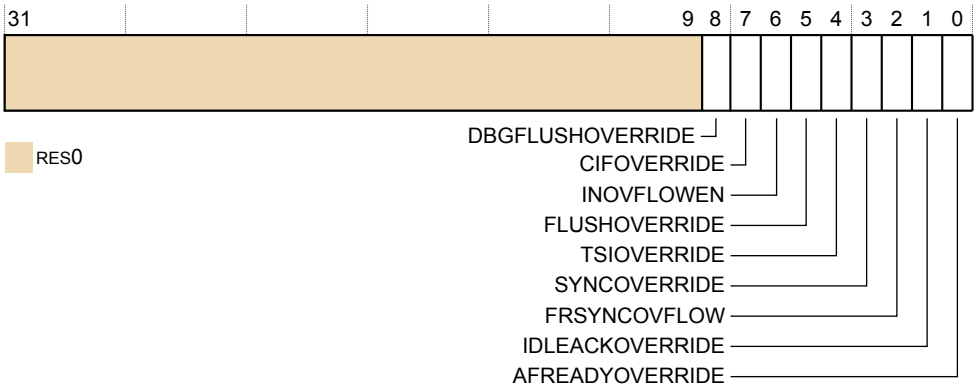

Figure D10-4 TRCAUXCTLR bit assignments

RES0, [31:9]

RES0 Reserved

DBGFLUSHOVERRIDE, [8]

Override trace flush on debug state entry. The possible values are:

0 Trace flush on debug state entry is enabled.

1 Trace flush on debug state entry is disabled.

CIFOVERRIDE, [7]

Override core interface register repeater clock enable. The possible values are:

0 Core interface clock gate is enabled.

1 Core interface clock gate is disabled.

INOVFLOWEN, [6]

Allow overflows of the core interface buffer, removing any rare impact that the trace unit might have on the core's speculation when enabled. The possible values are:

0 Core interface buffer overflows are disabled.

1 Core interface buffer overflows are enabled.

When this bit is set to 1, the trace start/stop logic might deviate from architecturally-specified behavior.

FLUSHOVERRIDE, [5]

Override ETM flush behavior. The possible values are:

0 ETM trace unit FIFO is flushed and ETM trace unit enters idle state when **DBGEN** or **NIDEN** is LOW.

1 ETM trace unit FIFO is not flushed and ETM trace unit does not enter idle state when **DBGEN** or **NIDEN** is LOW.

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

TSIOVERRIDE, [4]

Override TS packet insertion behavior. The possible values are:

`0` Timestamp packets are inserted into FIFO only when trace activity is LOW.

`1` Timestamp packets are inserted into FIFO irrespective of trace activity.

SYNCOVERRIDE, [3]

Override SYNC packet insertion behavior. The possible values are:

`0` SYNC packets are inserted into FIFO only when trace activity is low.

`1` SYNC packets are inserted into FIFO irrespective of trace activity.

FRSYNCOVFLOW, [2]

Force overflows to output synchronization packets. The possible values are:

`0` No FIFO overflow when SYNC packets are delayed.

`1` Forces FIFO overflow when SYNC packets are delayed.

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

IDLEACKOVERRIDE, [1]

Force ETM idle acknowledge. The possible values are:

`0` ETM trace unit idle acknowledge is asserted only when the ETM trace unit is in idle state.

`1` ETM trace unit idle acknowledge is asserted irrespective of the ETM trace unit idle state.

When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.

AFREADYOVERRIDE, [0]

Force assertion of **AFREADYM** output. The possible values are:

`0` ETM trace unit **AFREADYM** output is asserted only when the ETM trace unit is in idle state or when all the trace bytes in FIFO before a flush request are output.

`1` ETM trace unit **AFREADYM** output is always asserted HIGH.

When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.

The TRCAUXCTLR can be accessed through the internal memory-mapped interface and the external debug interface, offset `0x018`.

Configurations

Available in all configurations.

D10.6 TRCBBCTLR, Branch Broadcast Control Register

The TRCBBCTLR controls how branch broadcasting behaves, and allows branch broadcasting to be enabled for certain memory regions.

Bit field descriptions

The TRCAUXCTLR is a 32-bit register.

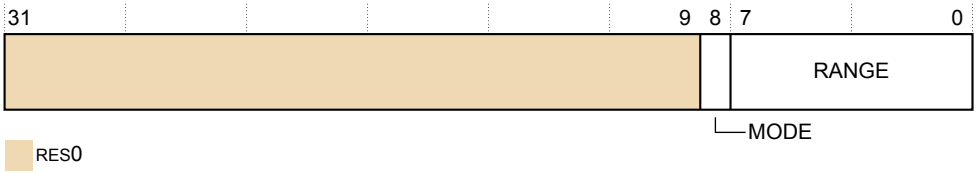

Figure D10-5 TRCBBCTLR bit assignments

RES0, [31:9]

RES0 Reserved

MODE, [8]

Mode bit:

`0` Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines.

If RANGE==0 then branch broadcasting is enabled for the entire memory map.

`1` Include mode. Branch broadcasting is enabled in the address range that RANGE defines.

If RANGE==0 then the behavior of the trace unit is CONSTRAINED UNPREDICTABLE. That is, the trace unit might or might not consider any instructions to be in a branch broadcast region.

RANGE, [7:0]

Address range field

Selects which address range comparator pairs are in use with branch broadcasting. Each bit represents an address range comparator pair, so bit[*n*] controls the selection of address range comparator pair *n*. If bit[*n*] is:

`0` The address range that address range comparator pair *n* defines, is not selected.

`1` The address range that address range comparator pair *n* defines, is selected.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCBBCTLR can be accessed through the external debug interface, offset `0x03C`.

D10.7 TRCCCCTLR, Cycle Count Control Register

The TRCCCCTLR sets the threshold value for cycle counting.

Bit field descriptions

The TRCCCCTLR is a 32-bit register.

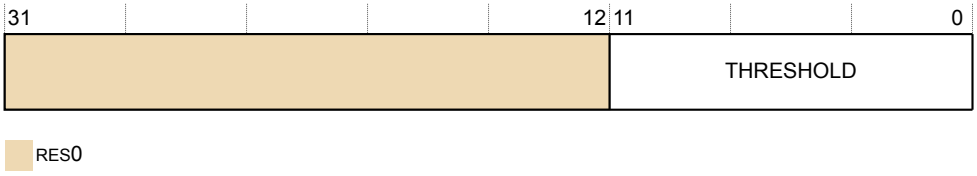

Figure D10-6 TRCCCCTLR bit assignments

RES0, [31:12]

RES0 Reserved

THRESHOLD, [11:0]

Instruction trace cycle count threshold

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCCCTLR can be accessed through the external debug interface, offset `0x038`.

D10.8 TRCCIDCCTLR0, Context ID Comparator Control Register 0

The TRCCIDCCTLR0 controls the mask value for the context ID comparators.

Bit field descriptions

The TRCCIDCCTLR0 is a 32-bit register.

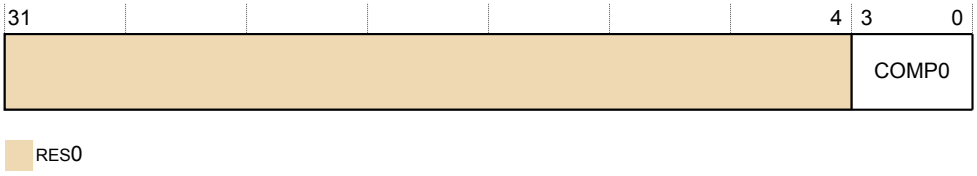

Figure D10-7 TRCCIDCCTLR0 bit assignments

RES0, [31:4]

RES0 Reserved

COMP0, [3:0]

Controls the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0. When a bit is:

`0` The trace unit includes the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison.

`1` The trace unit ignores the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDCCTLR0 can be accessed through the external debug interface, offset `0x680`.

D10.9 TRCCIDCVR0, Context ID Comparator Value Register 0

The TRCCIDCVR0 contains a Context ID value.

Bit field descriptions

The TRCCIDCVR0 is a 64-bit register.

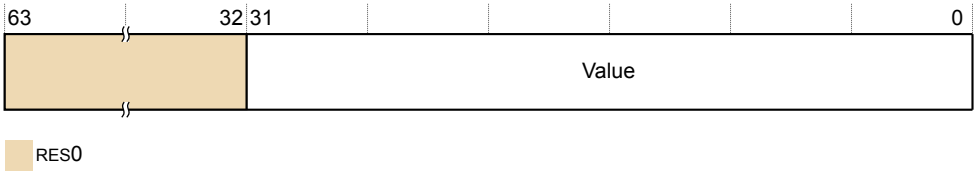

Figure D10-8 TRCCIDCVR0 bit assignments

RES0, [63:32]

RES0 Reserved

VALUE, [31:0]

The data value to compare against

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDCVR0 can be accessed through the external debug interface, offset `0x600`.

D10.10 TRCCIDR0, ETM Component Identification Register 0

The TRCCIDR0 provides information to identify a trace component.

Bit field descriptions

The TRCCIDR0 is a 32-bit register.

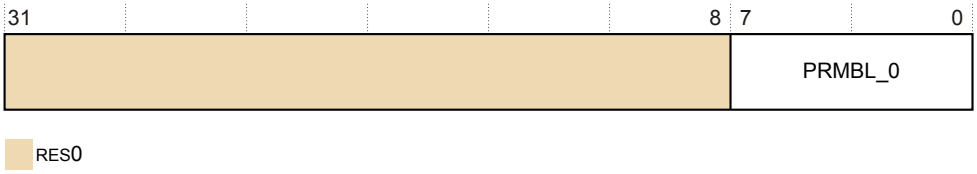

Figure D10-9 TRCCIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_0, [7:0]

`0x0D` Preamble byte 0

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDR0 can be accessed through the external debug interface, offset `0xFF0`.

D10.11 TRCCIDR1, ETM Component Identification Register 1

The TRCCIDR1 provides information to identify a trace component.

Bit field descriptions

The TRCCIDR1 is a 32-bit register.

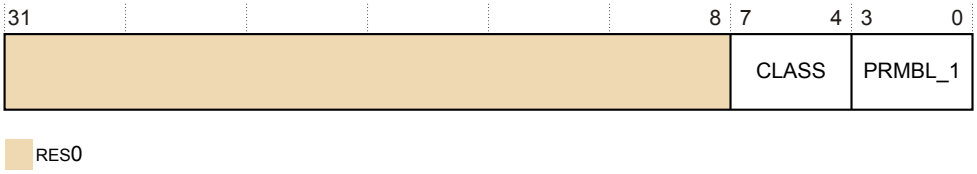

Figure D10-10 TRCCIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

CLASS, [7:4]

`0x9` Debug component

PRMBL_1, [3:0]

`0x0` Preamble byte 1

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDR1 can be accessed through the external debug interface, offset `0xFF4`.

D10.12 TRCCIDR2, ETM Component Identification Register 2

The TRCCIDR2 provides information to identify a *Cross Trigger Interface* (CTI) component.

Bit field descriptions

The TRCCIDR2 is a 32-bit register.

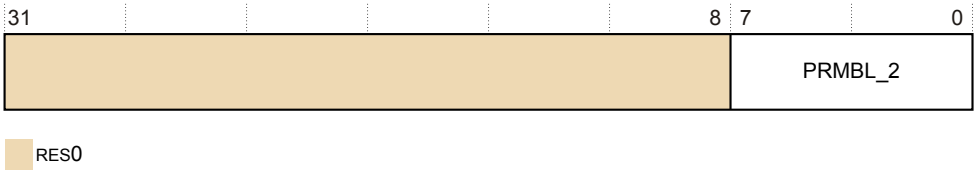

Figure D10-11 TRCCIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_2, [7:0]

`0x05` Preamble byte 2

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDR2 can be accessed through the external debug interface, offset `0xFF8`.

D10.13 TRCCIDR3, ETM Component Identification Register 3

The TRCCIDR3 provides information to identify a trace component.

Bit field descriptions

The TRCCIDR3 is a 32-bit register.

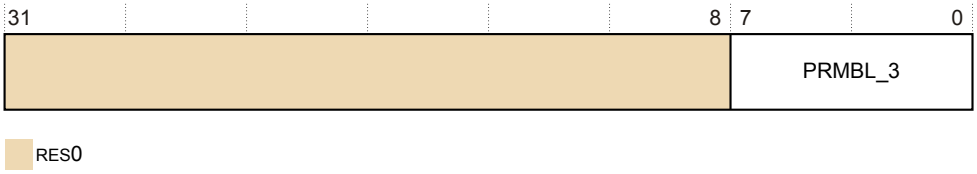

Figure D10-12 TRCCIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

PRMBL_3, [7:0]

`0xB1` Preamble byte 3

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCIDR3 can be accessed through the external debug interface, offset `0xFFC`.

D10.14 TRCCLAIMCLR, Claim Tag Clear Register

The TRCCLAIMCLR clears bits in the claim tag and determines the current value of the claim tag.

Bit field descriptions

The TRCCLAIMCLR is a 32-bit register.

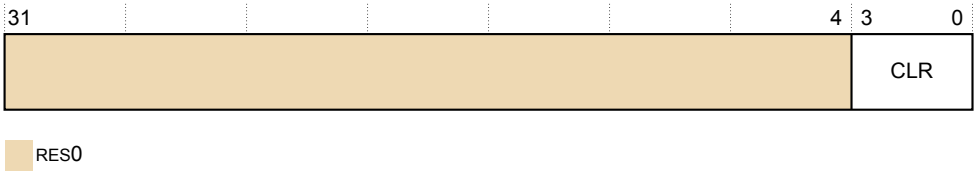

Figure D10-13 TRCCLAIMCLR bit assignments

RES0, [31:4]

RES0 Reserved

CLR, [3:0]

On reads, for each bit:

`0` Claim tag bit is not set.

`1` Claim tag bit is set.

On writes, for each bit:

`0` Has no effect.

`1` Clears the relevant bit of the claim tag.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCLAIMCLR can be accessed through the external debug interface, offset `0xFA4`.

D10.15 TRCCLAIMSET, Claim Tag Set Register

The TRCCLAIMSET sets bits in the claim tag and determines the number of claim tag bits implemented.

Bit field descriptions

The TRCCLAIMSET is a 32-bit register.

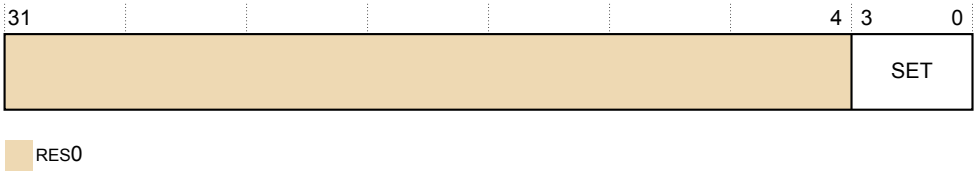

Figure D10-14 TRCCLAIMSET bit assignments

RES0, [31:4]

RES0 Reserved

SET, [3:0]

On reads, for each bit:

`0` Claim tag bit is not implemented.

`1` Claim tag bit is implemented.

On writes, for each bit:

`0` Has no effect.

`1` Sets the relevant bit of the claim tag.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCLAIMSET can be accessed through the external debug interface, offset `0xFA0`.

D10.16 TRCCNTCTLR0, Counter Control Register 0

The TRCCNTCTLR0 controls the counter.

Bit field descriptions

The TRCCNTCTLR0 is a 32-bit register.

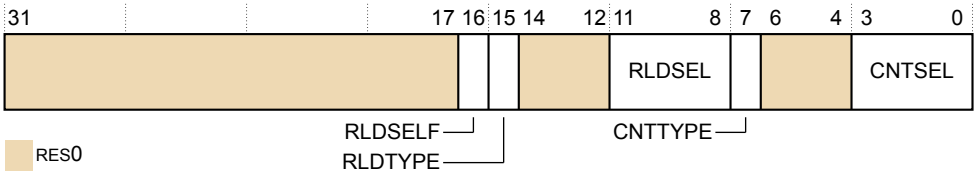

Figure D10-15 TRCCNTCTLR0 bit assignments

RES0, [31:17]

RES0 Reserved

RLDSELF, [16]

Defines whether the counter reloads when it reaches zero:

0 The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.

1 The counter reloads when it reaches zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL.

RLDTYPE, [15]

Selects the resource type for the reload:

0 Single selected resource

1 Boolean combined resource pair

RES0, [14:12]

RES0 Reserved

RLDSEL, [11:8]

Selects the resource number, based on the value of RLDTYPE:

When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

CNTTYPE, [7]

Selects the resource type for the counter:

0 Single selected resource

1 Boolean combined resource pair

RES0, [6:4]

RES0 Reserved

CNTSEL, [3:0]

Selects the resource number, based on the value of CNTTYPE:

When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTCTLR0 can be accessed through the external debug interface, offset `0x150`.

D10.17 TRCCNTCTLR1, Counter Control Register 1

The TRCCNTCTLR1 controls the counter.

Bit field descriptions

The TRCCNTCTLR1 is a 32-bit register.

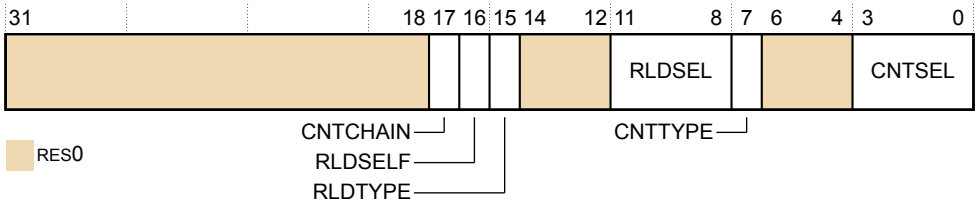

Figure D10-16 TRCCNTCTLR1 bit assignments

RES0, [31:18]

RES0 Reserved

CNTCHAIN, [17]

Defines whether the counter decrements when the counter reloads. This enables two counters to be used in combination to provide a larger counter:

0 The counter operates independently from the counter. The counter only decrements based on CNTTYPE and CNTSEL.

1 The counter decrements when the counter reloads. The counter also decrements when the resource selected by CNTTYPE and CNTSEL is active.

RLDSELF, [16]

Defines whether the counter reloads when it reaches zero:

0 The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.

1 The counter reloads when it is zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL.

RLDTYPE, [15]

Selects the resource type for the reload:

0 Single selected resource

1 Boolean combined resource pair

RES0, [14:12]

RES0 Reserved

RLDSEL, [11:8]

Selects the resource number, based on the value of RLDTYPE:

When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

CNTTYPE, [7]

Selects the resource type for the counter:

`0` Single selected resource

`1` Boolean combined resource pair

RES0, [6:4]

RES0 Reserved

CNTSEL, [3:0]

Selects the resource number, based on the value of CNTTYPE:

When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTCTLR1 can be accessed through the external debug interface, offset `0x154`.

D10.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1

The TRCCNTRLDVRn define the reload value for the counter.

Bit field descriptions

The TRCCNTRLDVRn is a 32-bit register.

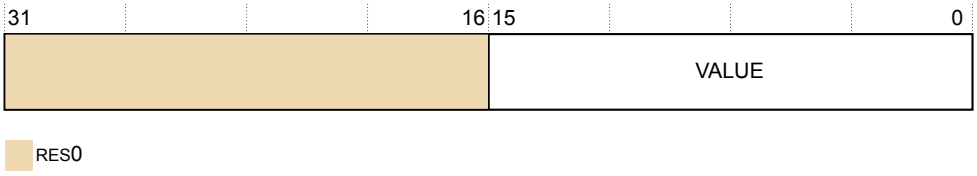

Figure D10-17 TRCCNTRLDVRn bit assignments

RES0, [31:16]

RES0 Reserved

VALUE, [15:0]

Defines the reload value for the counter. This value is loaded into the counter each time the reload event occurs.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTRLDVRn registers can be accessed through the external debug interface, offsets:

TRCCNTRLDVR0

`0x140`

TRCCNTRLDVR1

`0x144`

D10.19 TRCCNTVRn, Counter Value Registers 0-1

The TRCCNTVRn contain the current counter value.

Bit field descriptions

The TRCCNTVRn is a 32-bit register.

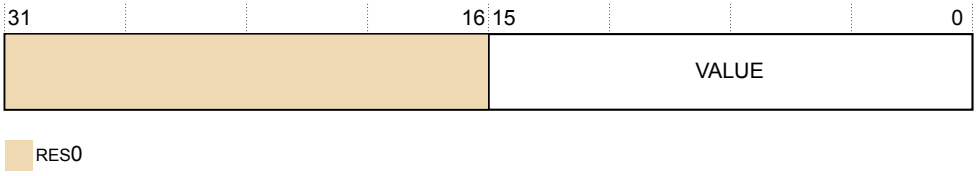

Figure D10-18 TRCCNTVRn bit assignments

RES0, [31:16]

RES0 Reserved

VALUE, [15:0]

Contains the current counter value.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCNTRLDVRn registers can be accessed through the external debug interface, offsets:

TRCCNTVR0

`0x160`

TRCCNTVR1

`0x164`

D10.20 TRCCONFIGR, Trace Configuration Register

The TRCCONFIGR controls the tracing options.

Bit field descriptions

The TRCCONFIGR is a 32-bit register.

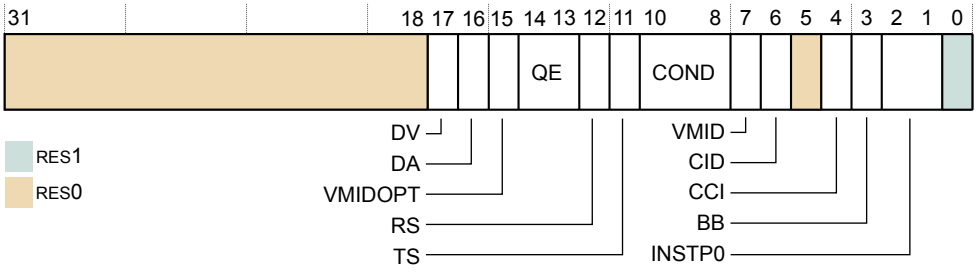

Figure D10-19 TRCCONFIGR bit assignments

RES0, [31:18]

RES0 Reserved

DV, [17]

Read-As-Zero (RAZ). Data value tracing is not supported.

DA, [16]

RAZ. Data address tracing is not supported.

VMIDOPT, [15]

Configures the Virtual context identifier value used by the trace unit, both for trace generation and in the Virtual context identifier comparators. The possible values are:

`0b0` VTTBR_EL2.VMID is used. If the trace unit supports a Virtual context identifier larger than the VTTBR_EL2.VMID, the upper unused bits are always zero. If the trace unit supports a Virtual context identifier larger than 8 bits and if the VTCR_EL2.VS bit forces use of an 8-bit Virtual context identifier, bits [15:8] of the trace unit Virtual context identifier are always zero.

`0b1` CONTEXTIDR_EL2 is used. TRCIDR2.VMIDOPT indicates whether this field is implemented.

QE, [14:13]

Enables Q element. The possible values are:

`0b00` Q elements are disabled.

`0b01` Q elements with instruction counts are disabled. Q elements without instruction counts are disabled.

`0b10` Reserved

`0b11` Q elements with and without instruction counts are enabled.

RS, [12]

Enables the return stack. The possible values are:

`0` Disables the return stack.

`1` Enables the return stack.

TS, [11]

Enables global timestamp tracing. The possible values are:

`0` Disables global timestamp tracing.

`1` Enables global timestamp tracing.

COND, [10:8]

Enables conditional instruction tracing. The possible 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.

VMID, [7]

Enables VMID tracing. The possible values are:

`0` Disables VMID tracing.

`1` Enables VMID tracing.

CID, [6]

Enables context ID tracing. The possible values are:

`0` Disables context ID tracing.

`1` Enables context ID tracing.

RES0, [5]

RES0 Reserved

CCI, [4]

Enables cycle counting instruction trace. The possible values are:

`0` Disables cycle counting instruction trace.

`1` Enables cycle counting instruction trace.

BB, [3]

Enables branch broadcast mode. The possible values are:

`0` Disables branch broadcast mode.

`1` Enables branch broadcast mode.

INSTP0, [2:1]

Controls whether load and store instructions are traced as P0 instructions. The possible values are:

`0b00` Load and store instructions are not traced as P0 instructions.

`0b01` Load instructions are traced as P0 instructions.

`0b10` Store instructions are traced as P0 instructions.

`0b11` Load and store instructions are traced as P0 instructions.

RES1, [0]

RES1 Reserved

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCCONFIGR can be accessed through the external debug interface, offset `0x010`.

D10.21 TRCDEVAFF0, Device Affinity Register 0

The TRCDEVAFF0 provides an additional core identification mechanism for scheduling purposes in a cluster. TRCDEVAFF0 is a read-only copy of MPIDR accessible from the external debug interface.

Bit field descriptions

The TRCDEVAFF0 is a 32-bit register and is a copy of the MPIDR_EL1[31:0]. See *B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314 for full bit field descriptions.

D10.22 TRCDEVAFF1, Device Affinity Register 1

The TRCDEVAFF1 is a read-only copy of MPIDR_EL1[63:32] as seen from EL3, unaffected by VMPIDR_EL2.

Bit field descriptions

See *B2.108 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B2-314 for full bit field descriptions.

D10.23 TRCDEVARCH, Device Architecture Register

The TRCDEVARCH identifies the ETM trace unit as an ETMv4 component.

Bit field descriptions

The TRCDEVARCH is a 32-bit register.

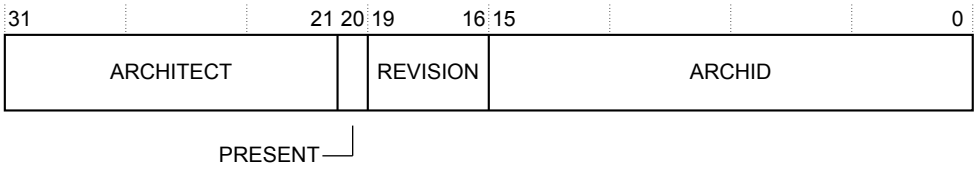

Figure D10-20 TRCDEVARCH bit assignments

ARCHITECT, [31:21]

Defines the architect of the component:

[31:28] `0x4`, Arm JEP continuation

[27:21] `0x3B`, Arm JEP 106 code

PRESENT, [20]

Indicates the presence of this register:

`0b1` Register is present.

REVISION, [19:16]

Architecture revision:

`0x02` Architecture revision 2

ARCHID, [15:0]

Architecture ID:

`0x4A13` ETMv4 component

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCDEVARCH can be accessed through the external debug interface, offset `0xFBC`.

D10.24 TRCDEVID, Device ID Register

The TRCDEVID indicates the capabilities of the ETM trace unit.

Bit field descriptions

The TRCDEVID is a 32-bit register.

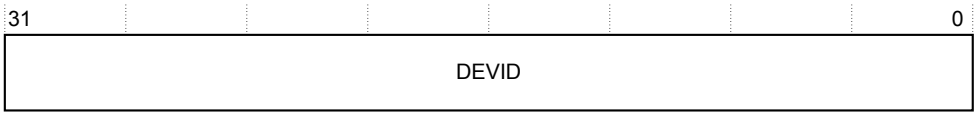

Figure D10-21 TRCDEVID bit assignments

DEVID, [31:0]

RAZ. There are no component-defined capabilities.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCDEVID can be accessed through the external debug interface, offset `0xFC8`.

D10.25 TRCDEVTYPE, Device Type Register

The TRCDEVTYPE indicates the type of the component.

Bit field descriptions

The TRCDEVTYPE is a 32-bit register.

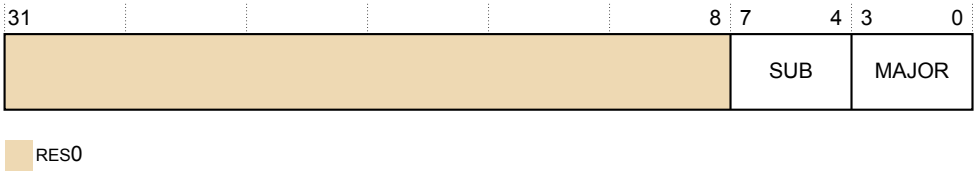

Figure D10-22 TRCDEVTYPE bit assignments

RES0, [31:8]

RES0 Reserved

SUB, [7:4]

The sub-type of the component:

`0b0001` Core trace

MAJOR, [3:0]

The main type of the component:

`0b0011` Trace source

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCDEVTYPE can be accessed through the external debug interface, offset `0xFCC`.

D10.26 TRCEVENTCTL0R, Event Control 0 Register

The TRCEVENTCTL0R controls the tracing of events in the trace stream. The events also drive the external outputs from the ETM trace unit. The events are selected from the Resource Selectors.

Bit field descriptions

The TRCEVENTCTL0R is a 32-bit register.

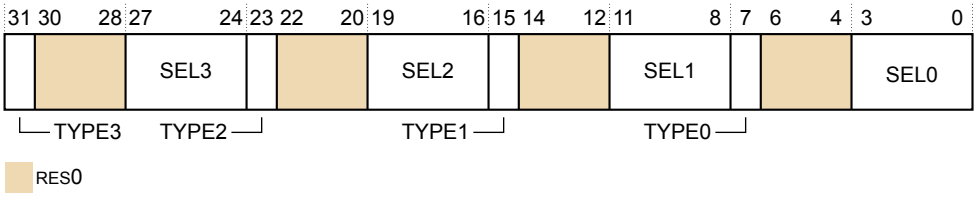

Figure D10-23 TRCEVENTCTL0R bit assignments

TYPE3, [31]

Selects the resource type for trace event 3:

0 Single selected resource

1 Boolean combined resource pair

RES0, [30:28]

RES0 Reserved

SEL3, [27:24]

Selects the resource number, based on the value of TYPE3:

When TYPE3 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE3 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TYPE2, [23]

Selects the resource type for trace event 2:

0 Single selected resource

1 Boolean combined resource pair

RES0, [22:20]

RES0 Reserved

SEL2, [19:16]

Selects the resource number, based on the value of TYPE2:

When TYPE2 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE2 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TYPE1, [15]

Selects the resource type for trace event 1:

0 Single selected resource

1 Boolean combined resource pair

RES0, [14:12]

RES0 Reserved

SEL1, [11:8]

Selects the resource number, based on the value of TYPE1:

When TYPE1 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TYPE0, [7]

Selects the resource type for trace event 0:

`0` Single selected resource

`1` Boolean combined resource pair

RES0, [6:4]

RES0 Reserved

SEL0, [3:0]

Selects the resource number, based on the value of TYPE0:

When TYPE0 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCEVENTCTL0R can be accessed through the external debug interface, offset `0x020`.

D10.27 TRCEVENTCTL1R, Event Control 1 Register

The TRCEVENTCTL1R controls the behavior of the events that TRCEVENTCTL0R selects.

Bit field descriptions

The TRCEVENTCTL1R is a 32-bit register.

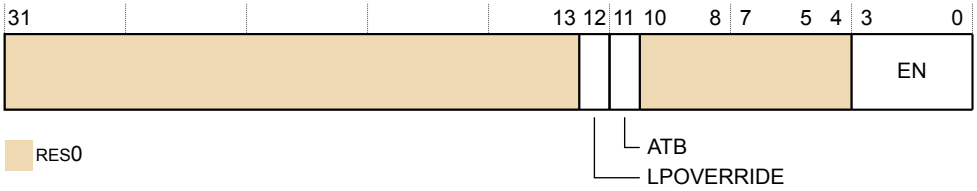

Figure D10-24 TRCEVENTCTL1R bit assignments

RES0, [31:13]

RES0 Reserved

LPOVERRIDE, [12]

Low-power state behavior override:

`0` Low-power state behavior unaffected

`1` Low-power state behavior overridden. The resources and Event trace generation are unaffected by entry to a low-power state.

ATB, [11]

ATB trigger enable:

`0` ATB trigger disabled

`1` ATB trigger enabled

RES0, [10:4]

RES0 Reserved

EN, [3:0]

One bit per event, to enable generation of an event element in the instruction trace stream when the selected event occurs:

`0` Event does not cause an event element.

`1` Event causes an event element.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCEVENTCTL1R can be accessed through the external debug interface, offset `0x024`.

D10.28 TRCEXTINSELR, External Input Select Register

The TRCEXTINSELR controls the selectors that choose an external input as a resource in the ETM trace unit. You can use the Resource Selectors to access these external input resources.

Bit field descriptions

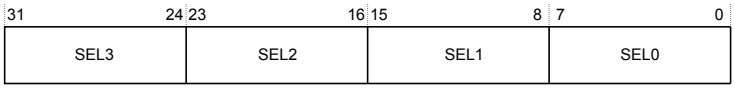

Figure D10-25 TRCEXTINSELR bit assignments

SEL3, [31:24]
Selects an event from the external input bus for External Input Resource 3.

SEL2, [23:16]
Selects an event from the external input bus for External Input Resource 2.

SEL1, [15:8]
Selects an event from the external input bus for External Input Resource 1.

SEL0, [7:0]
Selects an event from the external input bus for External Input Resource 0.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCEXTINSELR can be accessed through the external debug interface, offset `0x120`.

D10.29 TRCIDR0, ID Register 0

The TRCIDR0 returns the tracing capabilities of the ETM trace unit.

Bit field descriptions

The TRCIDR0 is a 32-bit register.

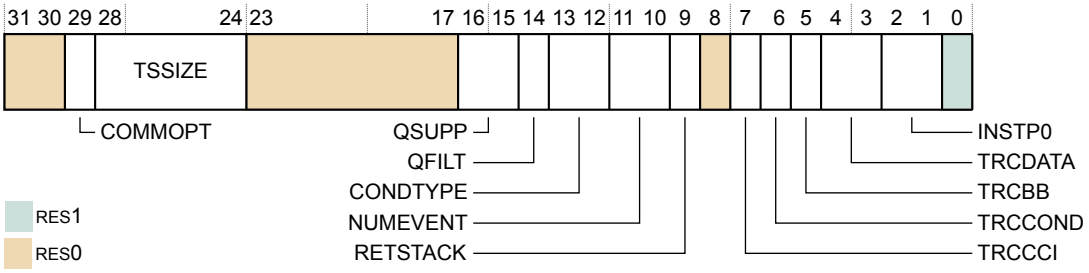

Figure D10-26 TRCIDR0 bit assignments

RES0, [31:30]

RES0 Reserved

COMMOPT, [29]

Indicates the meaning of the commit field in some packets:

`1` Commit mode 1

TSSIZE, [28:24]

Global timestamp size field:

`0b01000` Implementation supports a maximum global timestamp of 64 bits.

RES0, [23:17]

RES0 Reserved

QSUPP, [16:15]

Indicates Q element support:

`0b00` Q elements not supported

QFILT, [14]

Indicates Q element filtering support:

`0b0` Q element filtering not supported

CONDTYPE, [13:12]

Indicates how conditional results are traced:

`0b00` Conditional trace not supported

NUMEVENT, [11:10]

Number of events supported in the trace, minus 1:

`0b11` Four events supported

RETSTACK, [9]

Return stack support:

`1` Return stack implemented

RES0, [8]

RES0 Reserved

TRCCCI, [7]

Support for cycle counting in the instruction trace:

`1` Cycle counting in the instruction trace is implemented.

TRCCOND, [6]

Support for conditional instruction tracing:

`0` Conditional instruction tracing is not supported.

TRCBB, [5]

Support for branch broadcast tracing:

`1` Branch broadcast tracing is implemented.

TRCDATA, [4:3]

Conditional tracing field:

`0b00` Tracing of data addresses and data values is not implemented.

INSTP0, [2:1]

P0 tracing support field:

`0b00` Tracing of load and store instructions as P0 elements is not supported.

RES1, [0]

RES1 Reserved

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR0 can be accessed through the external debug interface, offset `0x1E0`.

D10.30 TRCIDR1, ID Register 1

The TRCIDR1 returns the base architecture of the trace unit.

Bit field descriptions

The TRCIDR1 is a 32-bit register.

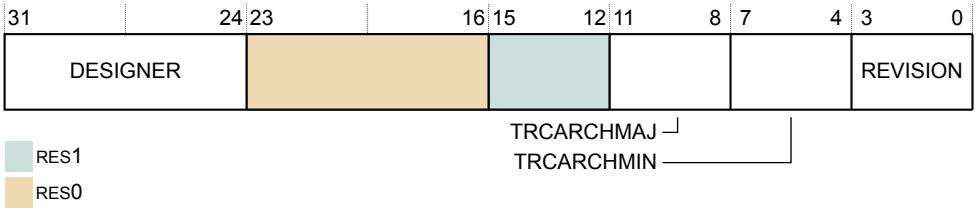

Figure D10-27 TRCIDR1 bit assignments

DESIGNER, [31:24]

Indicates which company designed the trace unit:

`0x41` Arm

RES0, [23:16]

RES0 Reserved

RES1, [15:12]

RES1 Reserved

TRCARCHMAJ, [11:8]

Major trace unit architecture version number:

`4` ETMv4

TRCARCHMIN, [7:4]

Minor trace unit architecture version number:

`0x2` ETMv4.2

REVISION, [3:0]

Trace unit implementation revision number:

`0x1` ETM revision for r0p1

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR1 can be accessed through the external debug interface, offset `0x1E4`.

D10.31 TRCIDR2, ID Register 2

The TRCIDR2 returns the maximum size of six parameters in the trace unit.

The parameters are:

- Cycle counter
- Data value
- Data address
- VMID
- Context ID
- Instruction address

Bit field descriptions

The TRCIDR2 is a 32-bit register.

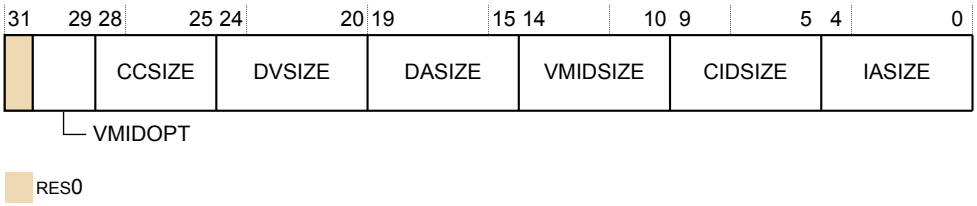

Figure D10-28 TRCIDR2 bit assignments

RES0, [31]

RES0 Reserved

VMIDOPT, [30:29]

Indicates the options for observing the Virtual context identifier:

`0x1` VMIDOPT implemented

CCSIZE, [28:25]

Size of the cycle counter in bits minus 12:

`0x0` The cycle counter is 12 bits in length.

DVSIZE, [24:20]

Data value size in bytes:

`0x00` Data value tracing is not implemented.

DASIZE, [19:15]

Data address size in bytes:

`0x00` Data address tracing is not implemented.

VMIDSIZE, [14:10]

Virtual Machine ID size:

`0x4` Maximum of 32-bit Virtual Machine ID size

CIDSIZE, [9:5]

Context ID size in bytes:

`0x4` Maximum of 32-bit Context ID size

IASIZE, [4:0]

Instruction address size in bytes:

`0x8` Maximum of 64-bit address size

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR2 can be accessed through the external debug interface, offset `0x1E8`.

D10.32 TRCIDR3, ID Register 3

The TRCIDR3 indicates:

- Whether TRCVICTLR is supported.
- The number of cores available for tracing.
- If an Exception level supports instruction tracing.
- The minimum threshold value for instruction trace cycle counting.
- Whether the synchronization period is fixed.
- Whether TRCSTALLCTLR is supported and if so whether it supports trace overflow prevention and supports stall control of the core.

Bit field descriptions

The TRCIDR3 is a 32-bit register.

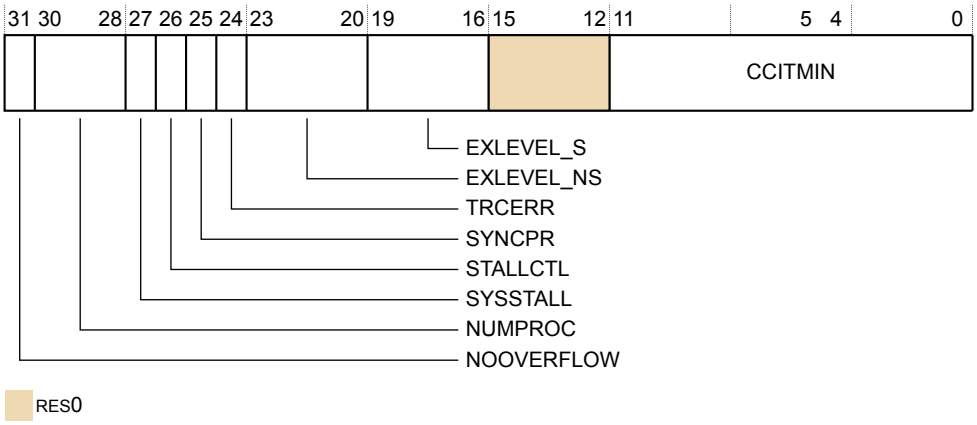

Figure D10-29 TRCIDR3 bit assignments

NOOVERFLOW, [31]

Indicates whether TRCSTALLCTLR.NOOVERFLOW is implemented:

`0` TRCSTALLCTLR.NOOVERFLOW is not implemented.

NUMPROC, [30:28]

Indicates the number of cores available for tracing:

`0b000` The trace unit can trace one core, ETM trace unit sharing not supported.

SYSSTALL, [27]

Indicates whether stall control is implemented:

`0` The system does not support core stall control.

STALLCTL, [26]

Indicates whether TRCSTALLCTLR is implemented:

`0` TRCSTALLCTLR is not implemented.

This field is used in conjunction with SYSSTALL.

SYNCPR, [25]

Indicates whether there is a fixed synchronization period:

`0` TRCSYNCPR is read-write so software can change the synchronization period.

TRCERR, [24]

Indicates whether TRCVICTLR.TRCERR is implemented:

`1` TRCVICTLR.TRCERR is implemented.

EXLEVEL_NS, [23:20]

Each bit controls whether instruction tracing in Non-secure state is implemented for the corresponding Exception level:

`0b0111` Instruction tracing is implemented for Non-secure EL0, EL1, and EL2 Exception levels.

EXLEVEL_S, [19:16]

Each bit controls whether instruction tracing in Secure state is implemented for the corresponding Exception level:

`0b1011` Instruction tracing is implemented for Secure EL0, EL1, and EL3 Exception levels.

RES0, [15:12]

RES0 Reserved

CCITMIN, [11:0]

The minimum value that can be programmed in TRCCCCTLR.THRESHOLD:

`0x004` Instruction trace cycle counting minimum threshold is 4.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR3 can be accessed through the external debug interface, offset `0x1EC`.

D10.33 TRCIDR4, ID Register 4

The TRCIDR4 indicates the resources available in the ETM trace unit.

Bit field descriptions

The TRCIDR4 is a 32-bit register.

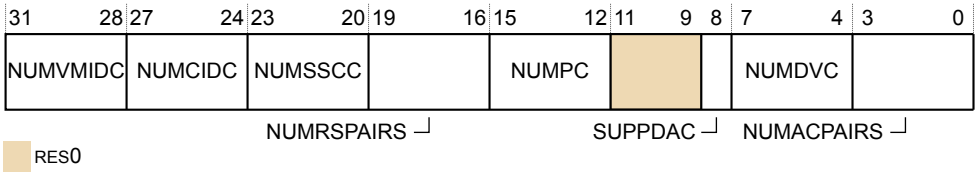

Figure D10-30 TRCIDR4 bit assignments

NUMVMIDC, [31:28]

Indicates the number of VMID comparators available for tracing:

`0x1` One VMID comparator is available.

NUMCIDC, [27:24]

Indicates the number of CID comparators available for tracing:

`0x1` One Context ID comparator is available.

NUMSSCC, [23:20]

Indicates the number of single-shot comparator controls available for tracing:

`0x1` One single-shot comparator control is available.

NUMRSPAIRS, [19:16]

Indicates the number of resource selection pairs available for tracing:

`0x7` Eight resource selection pairs are available.

NUMPC, [15:12]

Indicates the number of core comparator inputs available for tracing:

`0x0` Core comparator inputs are not implemented.

RES0, [11:9]

RES0 Reserved

SUPPDAC, [8]

Indicates whether the implementation supports data address comparisons: This value is:

`0` Data address comparisons are not implemented.

NUMDVC, [7:4]

Indicates the number of data value comparators available for tracing:

`0x0` Data value comparators not implemented.

NUMACPAIRS, [3:0]

Indicates the number of address comparator pairs available for tracing:

`0x4` Four address comparator pairs are implemented.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR4 can be accessed through the external debug interface, offset `0x1F0`.

D10.34 TRCIDR5, ID Register 5

The TRCIDR5 returns how many resources the trace unit supports.

Bit field descriptions

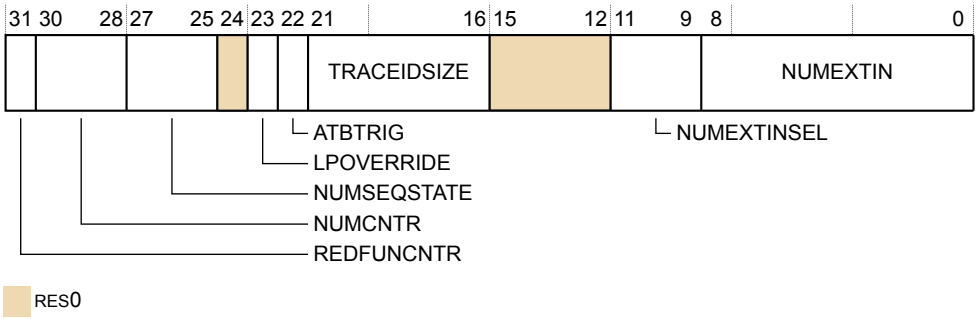

Figure D10-31 TRCIDR5 bit assignments

REDFUNCNTR, [31]

Reduced Function Counter implemented:

`0` Reduced Function Counter not implemented

NUMCNTR, [30:28]

Number of counters implemented:

`0b010` Two counters implemented

NUMSEQSTATE, [27:25]

Number of sequencer states implemented:

`0b100` Four sequencer states implemented

RES0, [24]

RES0 Reserved

LPOVERRIDE, [23]

Low-power state override support:

`0` Low-power state override support not implemented

ATBTRIG, [22]

ATB trigger support:

`1` ATB trigger support implemented

TRACEIDSIZE, [21:16]

Number of bits of trace ID:

`0x07` Seven-bit trace ID implemented

RES0, [15:12]

RES0 Reserved

NUMEXTINSEL, [11:9]

Number of external input selectors implemented:

`0b100` Four external input selectors implemented

NUMEXTIN, [8:0]

Number of external inputs implemented:

`0xD6` 32 external inputs implemented

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR5 can be accessed through the external debug interface, offset `0x1F4`.

D10.35 TRCIDR8, ID Register 8

The TRCIDR8 returns the maximum speculation depth of the instruction trace stream.

Bit field descriptions

The TRCIDR8 is a 32-bit register.

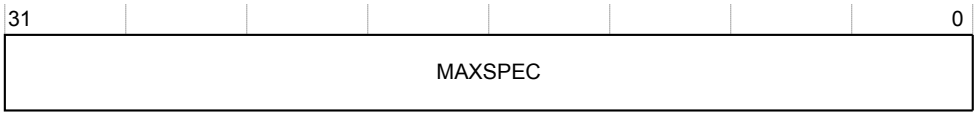

Figure D10-32 TRCIDR8 bit assignments

MAXSPEC, [31:0]

The maximum number of P0 elements in the trace stream that can be speculative at any time.

`0` Maximum speculation depth of the instruction trace stream

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR8 can be accessed through the external debug interface, offset `0x180`.

D10.36 TRCIDR9, ID Register 9

The TRCIDR9 returns the number of P0 right-hand keys that the trace unit can use.

Bit field descriptions

The TRCIDR9 is a 32-bit register.

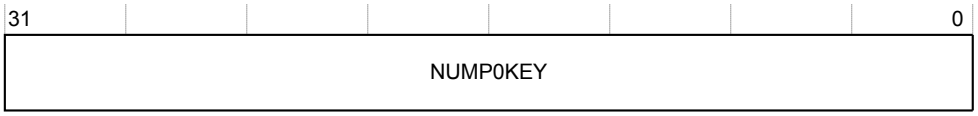

Figure D10-33 TRCIDR9 bit assignments

NUMP0KEY, [31:0]

The number of P0 right-hand keys that the trace unit can use.

`0` Number of P0 right-hand keys

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR9 can be accessed through the external debug interface, offset `0x184`.

D10.37 TRCIDR10, ID Register 10

The TRCIDR10 returns the number of P1 right-hand keys that the trace unit can use.

Bit field descriptions

The TRCIDR10 is a 32-bit register.

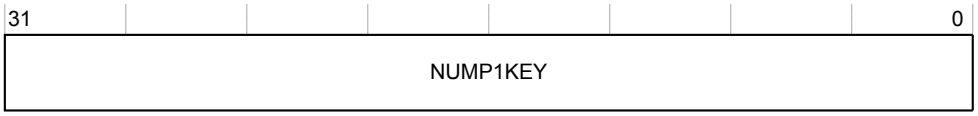

Figure D10-34 TRCIDR10 bit assignments

NUMP1KEY, [31:0]

The number of P1 right-hand keys that the trace unit can use.

`0` Number of P1 right-hand keys

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR10 can be accessed through the external debug interface, offset `0x188`.

D10.38 TRCIDR11, ID Register 11

The TRCIDR11 returns the number of special P1 right-hand keys that the trace unit can use.

Bit field descriptions

The TRCIDR11 is a 32-bit register.

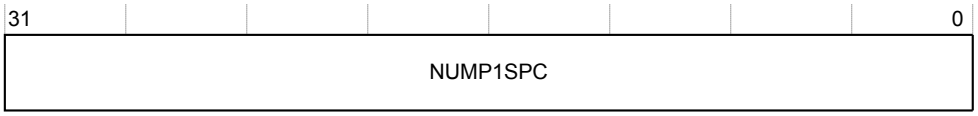

Figure D10-35 TRCIDR11 bit assignments

NUMP1SPC, [31:0]

The number of special P1 right-hand keys that the trace unit can use.

`0` Number of special P1 right-hand keys

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR11 can be accessed through the external debug interface, offset `0x18C`.

D10.39 TRCIDR12, ID Register 12

The TRCIDR12 returns the number of conditional instruction right-hand keys that the trace unit can use.

Bit field descriptions

The TRCIDR10 is a 32-bit register.

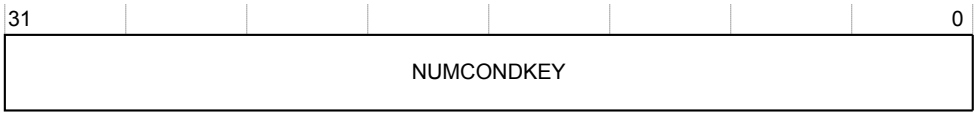

Figure D10-36 TRCIDR12 bit assignments

NUMCONDKEY, [31:0]

The number of conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

`0` Number of conditional instruction right-hand keys

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR12 can be accessed through the external debug interface, offset `0x190`.

D10.40 TRCIDR13, ID Register 13

The TRCIDR13 returns the number of special conditional instruction right-hand keys that the trace unit can use.

Bit field descriptions

The TRCIDR11 is a 32-bit register.

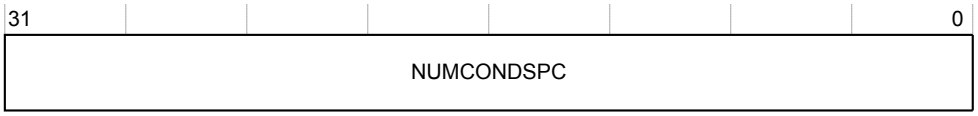

Figure D10-37 TRCIDR13 bit assignments

NUMCONDSPC, [31:0]

The number of special conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

`0` Number of special conditional instruction right-hand keys

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIDR13 can be accessed through the external debug interface, offset `0x194`.

D10.41 TRCIMSPEC0, IMPLEMENTATION SPECIFIC Register 0

The TRCIMSPEC0 shows the presence of any IMPLEMENTATION SPECIFIC features, and enables any features that are provided.

Bit field descriptions

The TRCIMSPEC0 is a 32-bit register.

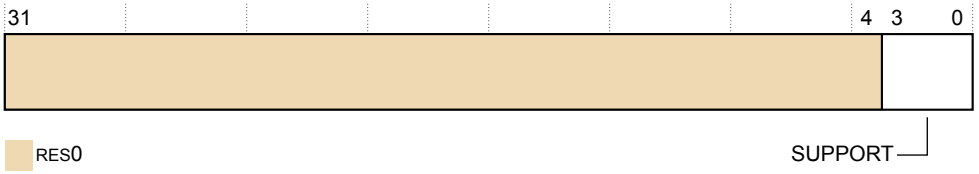

Figure D10-38 TRCIMSPEC0 bit assignments

RES0, [31:4]

RES0 Reserved

SUPPORT, [3:0]

`0` No IMPLEMENTATION SPECIFIC extensions supported

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCIMSPEC0 can be accessed through the external debug interface, offset `0x1C0`.

Note

System register accesses to the TRCIMSPEC0 will result in an UNDEFINED exception.

D10.42 TRCITATBCTR0, Trace Integration Test ATB Control Register 0

TRCITATBCTR0 controls signal outputs when **TRCITCTRL.IME** is set.

Bit field descriptions

The TRCITATBCTR0 is a 32-bit register.

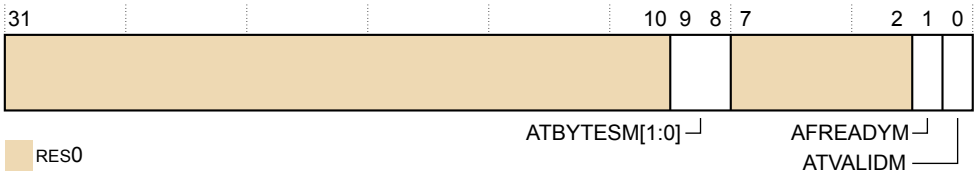

Figure D10-39 TRCITATBCTR0 bit assignments

ATBYTESM[1:0], [9:8]
Drives the **ATBYTESM** outputs.

AFREADYM, [1]
Drives the **AFREADYM** output.

ATVALIDM, [0]
Drives the **ATVALIDM** output.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITATBCTR0 register can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xEF8`.

D10.43 TRCITATBCTR1, Trace Integration Test ATB Control Register 1

TRCITATBCTR1 controls the **ATIDM[6:0]** signals when TRCITCTRL.IME is set.

Bit field descriptions

The TRCITATBCTR1 is a 32-bit register.

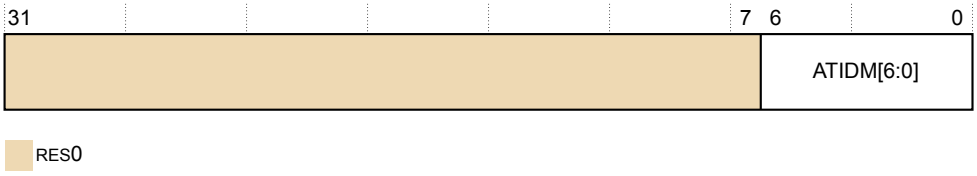

Figure D10-40 TRCITATBCTR1 bit assignments

ATIDM[6:0], [6:0]
Drives the **ATIDM[6:0]** outputs.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITATBCTR1 register can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xEF4`.

D10.44 TRCITATBCTR2, Trace Integration Test ATB Control Register 2

TRCITATBCTR2 enables the values of signal inputs to be read when bit[0] of the Integration Mode Control Register is set.

Bit field descriptions

The TRCITATBCTR2 is a 32-bit register.

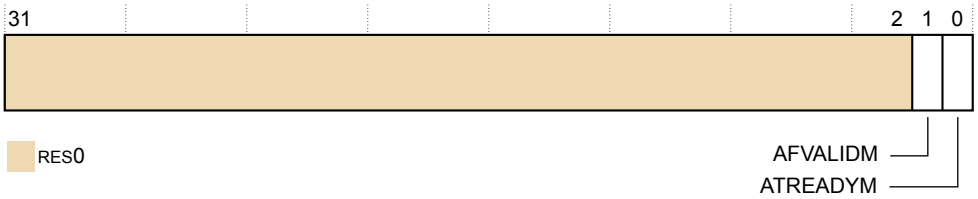

Figure D10-41 TRCITATBCTR2 bit assignments

AFVALIDM, [1]
Returns the value of **AFVALIDM** input.

ATREADYM, [0]
Returns the value of **ATREADYM** input. To sample **ATREADYM** correctly from the processor signals, **ATVALIDM** must be asserted.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITATBCTR2 register can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xEF0`.

D10.45 TRCITATBDATA0, Trace Integration Test ATB Data Register 0

TRCITATBDATA0 controls signal outputs when **TRCITCTRL.IME** is set.

Bit field descriptions

The TRCITATBDATA0 is a 32-bit register.

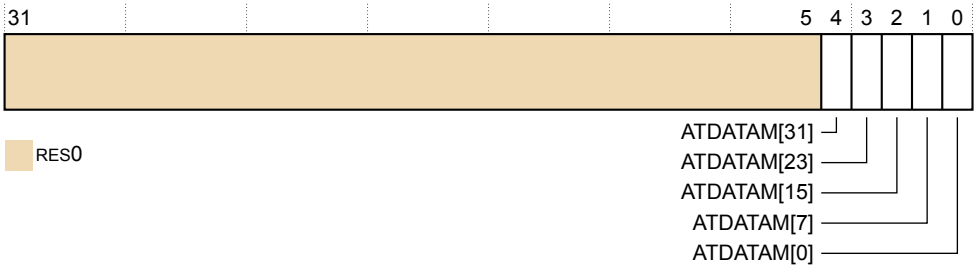

Figure D10-42 TRCITATBDATA0 bit assignments

ATDATAM[31], [4] Drives the **ATDATAM[31]** output.

ATDATAM[23], [3] Drives the **ATDATAM[23]** output.

ATDATAM[15], [2] Drives the **ATDATAM[15]** output.

ATDATAM[7], [1] Drives the **ATDATAM[7]** output.

ATDATAM[0], [0] Drives the **ATDATAM[0]** output.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITATBDATA0 register can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xEEC`.

D10.46 TRCITCTRL, Trace Integration Mode Control register

TRCITCTRL controls whether the trace unit is in integration mode.

Bit field descriptions

The TRCITCTRL is a 32-bit RW management register that is reset to zero.

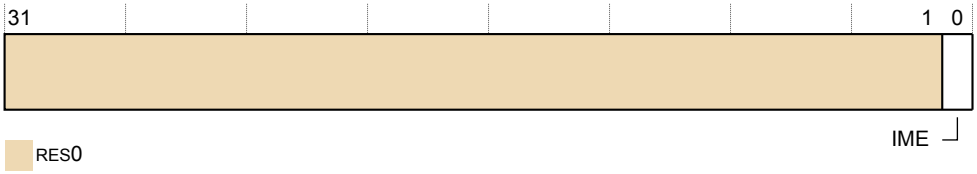

Figure D10-43 TRCITCTRL bit assignments

IME, [0]

Integration mode enable bit. The possible values are:

`0b0` The trace unit is not in integration mode.

`0b1` The trace unit is in integration mode. This mode enables:

- A debug agent to perform topology detection.
- SoC test software to perform integration testing.

Usage constraints

- Accessible only from the memory-mapped interface or from an external agent such as a debugger.
- If the IME bit changes from one to zero then Arm recommends that the trace unit is reset. Otherwise the trace unit might generate incorrect or corrupt trace and the trace unit resources might behave unexpectedly.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITCTRL register can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xF00`.

D10.47 TRCITMISCIN, Trace Integration Miscellaneous Input Register

TRCITMISCIN enables the values of signal inputs to be read when **TRCITCTRL.IME** is set.

Bit field descriptions

The TRCITMISCIN is a 32-bit register.

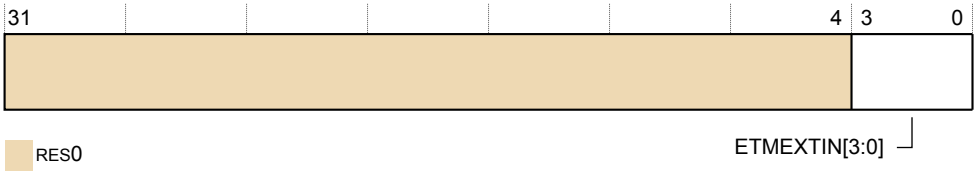

Figure D10-44 TRCITMISCIN bit assignments

ETMEXTIN[3:0], [3:0]
Returns the value of the **ETMEXTIN[3:0]** inputs.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITMISCIN register can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xEE0`.

D10.48 TRCITMISCOUT, Trace Integration Miscellaneous Outputs Register

TRCITMISCOUT controls signal outputs when **TRCITCTRL.IME** is set.

Bit field descriptions

The TRCITMISCOUT is a 32-bit register.

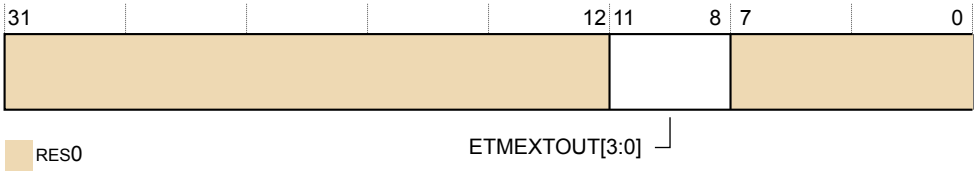

Figure D10-45 TRCITMISCOUT bit assignments

ETMEXTOUT[3:0], [11:8]
Drives the **EXTOUT[3:0]** outputs.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCITMISCOUT register can be accessed through the internal memory-mapped interface and the external debug interface, offset `0xEDC`.

D10.49 TRCLAR, Software Lock Access Register

The TRCLAR controls access to registers using the memory-mapped interface, when **PADDRDBG31** is LOW.

Bit field descriptions

The TRCLAR is a 32-bit register.

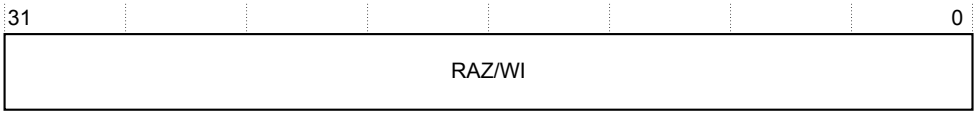

Figure D10-46 TRCLAR bit assignments

RAZ/WI, [31:0]

Read-As-Zero, write ignore

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCLAR can be accessed through the external debug interface, offset `0xFB0`.

D10.50 TRCLSR, Software Lock Status Register

The TRCLSR determines whether the software lock is implemented, and indicates the current status of the software lock.

Bit field descriptions

The TRCLSR is a 32-bit register.

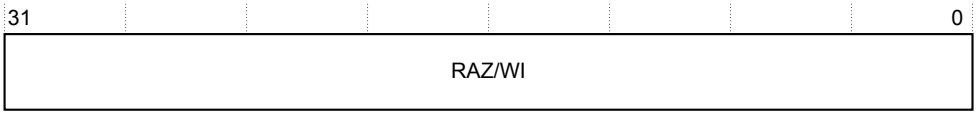

Figure D10-47 TRCLSR bit assignments

RAZ/WI, [31:0]
Read-As-Zero, write ignore

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCLSR can be accessed through the external debug interface, offset `0xFB4`.

D10.51 TRCOSLAR, OS Lock Access Register

The TRCOSLAR sets and clears the OS Lock, to lock out external debugger accesses to the ETM trace unit registers.

Bit field descriptions

The TRCOSLAR is a 32-bit register.

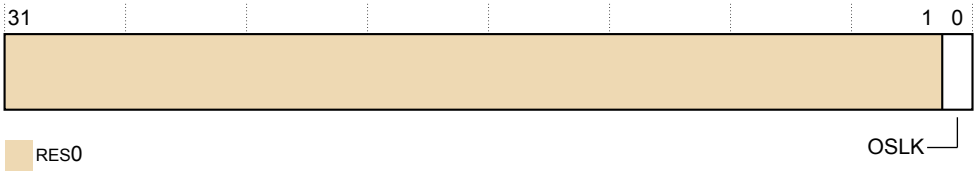

Figure D10-48 TRCOSLAR bit assignments

RES0, [31:1]

RES0 Reserved

OSLK, [0]

OS Lock key value:

`0` Unlock the OS Lock

`1` Lock the OS Lock

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCOSLAR can be accessed through the external debug interface, offset `0x300`.

D10.52 TRCOSLSR, OS Lock Status Register

The TRCOSLSR returns the status of the OS Lock.

Bit field descriptions

The TRCOSLSR is a 32-bit register.

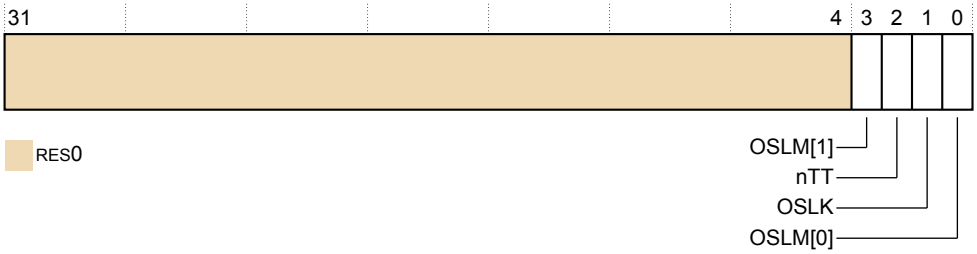

Figure D10-49 TRCOSLSR bit assignments

RES0, [31:4]

RES0 Reserved

OSLM[1], [3]

OS Lock model [1] bit. This bit is combined with OSLM[0] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always `0b10`, indicating that the OS Lock is implemented.

nTT, [2]

This bit is RAZ, that indicates that software must perform a 32-bit write to update the TRCOSLAR.

OSLK, [1]

OS Lock status bit:

`0` OS Lock is unlocked.

`1` OS Lock is locked.

OSLM[0], [0]

OS Lock model [0] bit. This bit is combined with OSLM[1] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always `0b10`, indicating that the OS Lock is implemented.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCOSLSR can be accessed through the external debug interface, offset `0x304`.

D10.53 TRCPDCR, Power Down Control Register

The TRCPDCR request to the system power controller to keep the ETM trace unit powered up.

Bit field descriptions

The TRCPDCR is a 32-bit register.

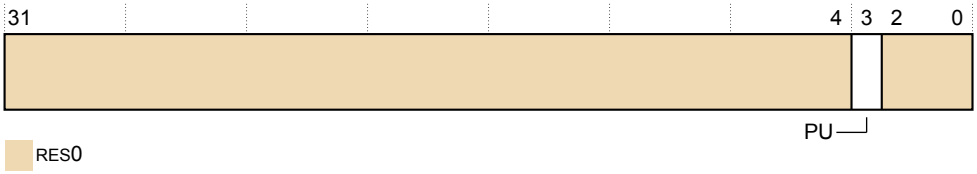

Figure D10-50 TRCPDCR bit assignments

RES0, [31:4]

RES0 Reserved

PU, [3]

Powerup request, to request that power to the ETM trace unit and access to the trace registers is maintained:

`0` Power not requested

`1` Power requested

This bit is reset to 0 on a trace unit reset.

RES0, [2:0]

RES0 Reserved

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPDCR can be accessed through the external debug interface, offset `0x310`.

D10.54 TRCPDSR, Power Down Status Register

The TRCPDSR indicates the power down status of the ETM trace unit.

Bit field descriptions

The TRCPDSR is a 32-bit register.

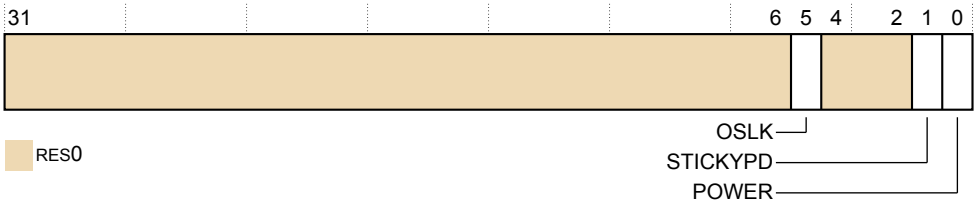

Figure D10-51 TRCPDSR bit assignments

RES0, [31:6]

RES0 Reserved

OSLK, [5]

OS lock status

`0` The OS Lock is unlocked.

`1` The OS Lock is locked.

RES0, [4:2]

RES0 Reserved

STICKYPD, [1]

Sticky power down state

`0` Trace register power has not been removed since the TRCPDSR was last read.

`1` Trace register power has been removed since the TRCPDSR was last read.

This bit is set to 1 when power to the ETM trace unit registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.

POWER, [0]

Indicates the ETM trace unit is powered:

`0` ETM trace unit is not powered. The trace registers are not accessible and they all return an error response.

`1` ETM trace unit is powered. All registers are accessible.

If a system implementation allows the ETM trace unit to be powered off independently of the debug power domain, the system must handle accesses to the ETM trace unit appropriately.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPDSR can be accessed through the external debug interface, offset `0x314`.

D10.55 TRCPIDR0, ETM Peripheral Identification Register 0

The TRCPIDR0 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR0 is a 32-bit register.

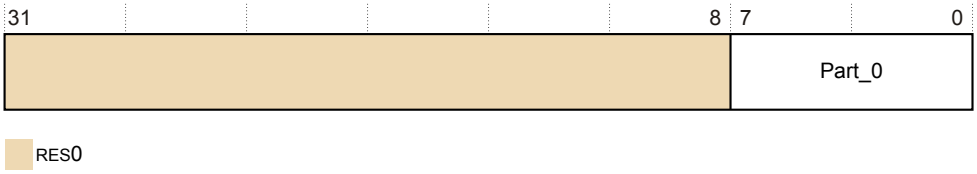

Figure D10-52 TRCPIDR0 bit assignments

RES0, [31:8]

RES0 Reserved

Part_0, [7:0]

`0x4B` Least significant byte of the ETM trace unit part number

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR0 can be accessed through the external debug interface, offset `0xFE0`.

D10.56 TRCPIDR1, ETM Peripheral Identification Register 1

The TRCPIDR1 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR1 is a 32-bit register.

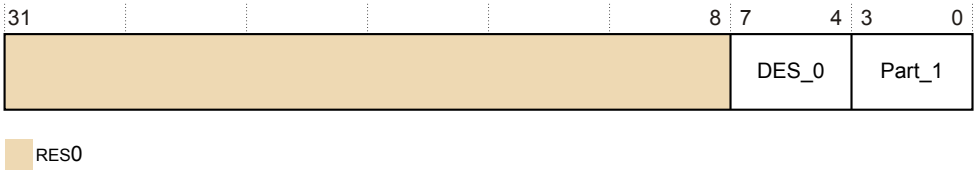

Figure D10-53 TRCPIDR1 bit assignments

RES0, [31:8]

RES0 Reserved

DES_0, [7:4]

`0xB` Arm Limited. This is bits[3:0] of JEP106 ID code.

Part_1, [3:0]

`0xD` Most significant four bits of the ETM trace unit part number

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR1 can be accessed through the external debug interface, offset `0xFE4`.

D10.57 TRCPIDR2, ETM Peripheral Identification Register 2

The TRCPIDR2 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR2 is a 32-bit register.

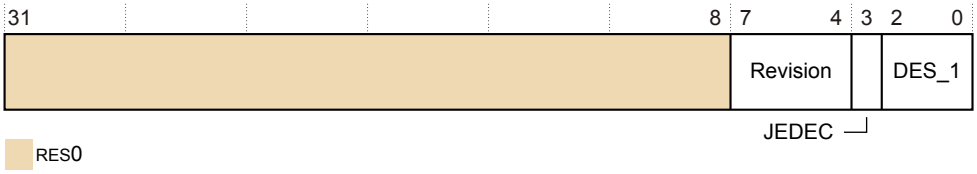

Figure D10-54 TRCPIDR2 bit assignments

RES0, [31:8]

RES0 Reserved

Revision, [7:4]

`0x1` r0p1

JEDEC, [3]

`0b1` RES1. Indicates a JEP106 identity code is used.

DES_1, [2:0]

`0b011` Arm Limited. This is bits[6:4] of JEP106 ID code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR2 can be accessed through the external debug interface, offset `0xFE8`.

D10.58 TRCPIDR3, ETM Peripheral Identification Register 3

The TRCPIDR3 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR3 is a 32-bit register.

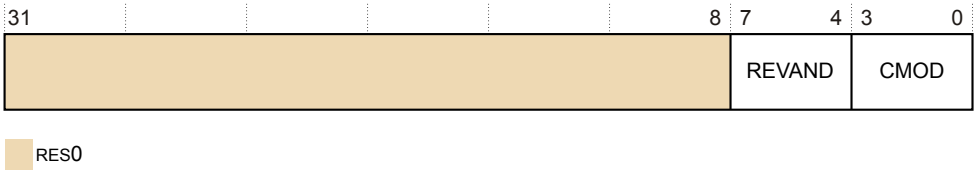

Figure D10-55 TRCPIDR3 bit assignments

RES0, [31:8]

RES0 Reserved

REVAND, [7:4]

`0x0` Part minor revision

CMOD, [3:0]

`0x0` Not customer modified

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR3 can be accessed through the external debug interface, offset `0xFEC`.

D10.59 TRCPIDR4, ETM Peripheral Identification Register 4

The TRCPIDR4 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR4 is a 32-bit register.

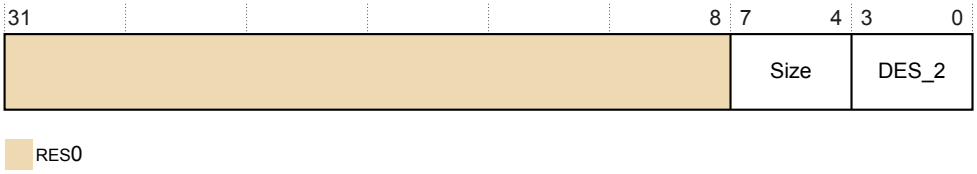

Figure D10-56 TRCPIDR4 bit assignments

RES0, [31:8]

RES0 Reserved

Size, [7:4]

`0x0` Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

`0x4` Arm Limited. This is bits[3:0] of the JEP106 continuation code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPIDR4 can be accessed through the external debug interface, offset `0xFD0`.

D10.60 TRCPIDRn, ETM Peripheral Identification Registers 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers.

They are reserved for future use and are RES0.

D10.61 TRCPRGCTLR, Programming Control Register

The TRCPRGCTLR enables the ETM trace unit.

Bit field descriptions

The TRCPRGCTLR is a 32-bit register.

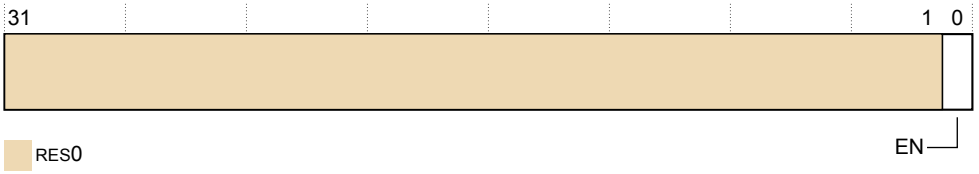

Figure D10-57 TRCPRGCTLR bit assignments

RES0, [31:1]

RES0 Reserved

EN, [0]

Trace program enable:

`0` The ETM trace unit interface in the core is disabled, and clocks are enabled only when necessary to process APB accesses, or drain any already generated trace. This is the reset value.

`1` The ETM trace unit interface in the core is enabled, and clocks are enabled. Writes to most trace registers are IGNORED.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCPRGCTLR can be accessed through the external debug interface, offset `0x004`.

D10.62 TRCRSCTLRn, Resource Selection Control Registers 2-16

The TRCRSCTLRn controls the trace resources. There are eight resource pairs, the first pair is predefined as {0,1,pair=0} and having reserved select registers. This leaves seven pairs to be implemented as programmable selectors.

Bit field descriptions

The TRCRSCTLRn is a 32-bit register.

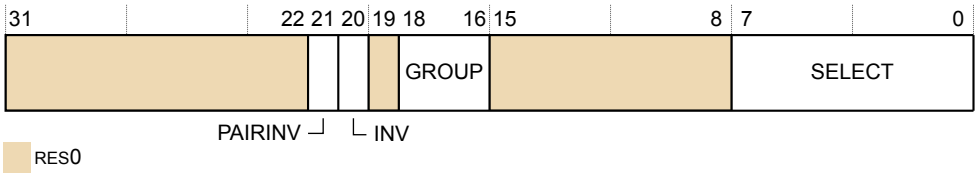

Figure D10-58 TRCRSCTLRn bit assignments

RES0, [31:22]

RES0 Reserved

PAIRINV, [21]

Inverts the result of a combined pair of resources.

This bit is implemented only on the lower register for a pair of resource selectors.

INV, [20]

Inverts the selected resources:

`0` Resource is not inverted.

`1` Resource is inverted.

RES0, [19]

RES0 Reserved

GROUP, [18:16]

Selects a group of resources. See the *Arm® ETM Architecture Specification, ETMv4* for more information.

RES0, [15:8]

RES0 Reserved

SELECT, [7:0]

Selects one or more resources from the required group. One bit is provided for each resource from the group.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCRSCTLRn can be accessed through the external debug interface, offset `0x208-0x023C`.

D10.63 TRCSEQEVRn, Sequencer State Transition Control Registers 0-2

The TRCSEQEVRn defines the sequencer transitions that progress to the next state or backwards to the previous state. The ETM trace unit implements a sequencer state machine with up to four states.

Bit field descriptions

The TRCSEQEVRn is a 32-bit register.

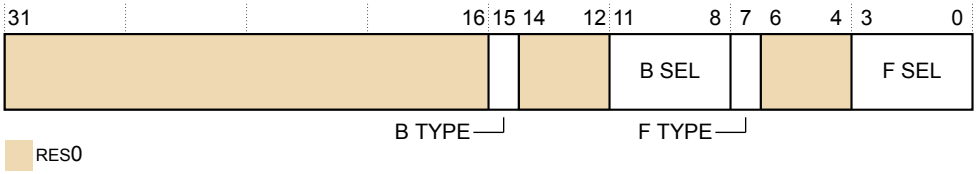

Figure D10-59 TRCSEQEVRn bit assignments

RES0, [31:16]

RES0 Reserved

B TYPE, [15]

Selects the resource type to move backwards to this state from the next state:

0 Single selected resource

1 Boolean combined resource pair

RES0, [14:12]

RES0 Reserved

B SEL, [11:8]

Selects the resource number, based on the value of B TYPE:

When B TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When B TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

F TYPE, [7]

Selects the resource type to move forwards from this state to the next state:

0 Single selected resource

1 Boolean combined resource pair

RES0, [6:4]

RES0 Reserved

F SEL, [3:0]

Selects the resource number, based on the value of F TYPE:

When F TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When F TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSEQEVRn registers can be accessed through the external debug interface, offsets:

TRCSEQEVR0
0x100

TRCSEQEVR1
0x104

TRCSEQEVR2
0x108

D10.64 TRCSEQRSTEVR, Sequencer Reset Control Register

The TRCSEQRSTEVR resets the sequencer to state 0.

Bit field descriptions

The TRCSEQRSTEVR is a 32-bit register

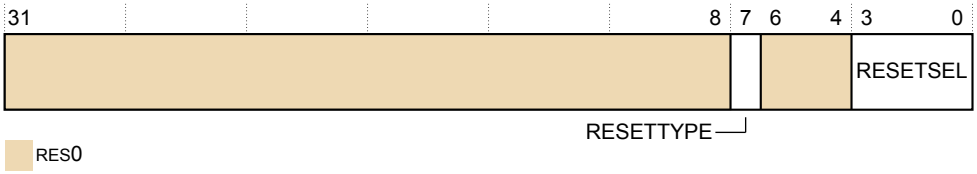

Figure D10-60 TRCSEQRSTEVR bit assignments

RES0, [31:8]

RES0 Reserved

RESETTYPE, [7]

Selects the resource type to move back to state 0:

`0` Single selected resource

`1` Boolean combined resource pair

RES0, [6:4]

RES0 Reserved

RESETSEL, [3:0]

Selects the resource number, based on the value of RESETTYPE:

When RESETTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When RESETTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSEQRSTEVR can be accessed through the external debug interface, offset `0x118`.

D10.65 TRCSEQSTR, Sequencer State Register

The TRCSEQSTR holds the value of the current state of the sequencer.

Bit field descriptions

The TRCSEQSTR is a 32-bit register

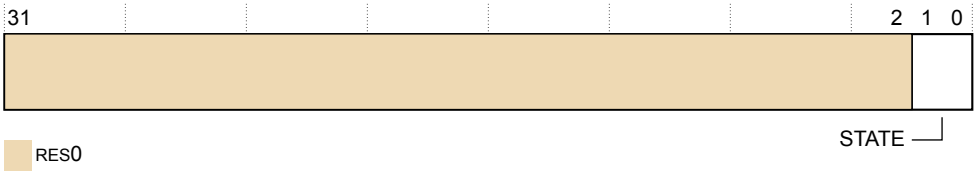

Figure D10-61 TRCSEQSTR bit assignments

RES0, [31:2]

RES0 Reserved

STATE, [1:0]

Current sequencer state:

`0b00` State 0
`0b01` State 1
`0b10` State 2
`0b11` State 3

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSEQSTR can be accessed through the external debug interface, offset `0x11C`.

D10.66 TRCSSCCR0, Single-Shot Comparator Control Register 0

The TRCSSCCR0 controls the single-shot comparator.

Bit field descriptions

The TRCSSCSR0 is a 32-bit register

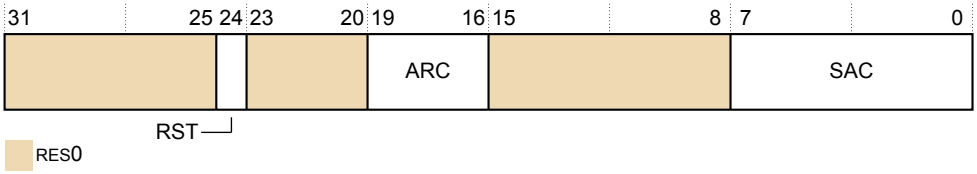

Figure D10-62 TRCSSCCR0 bit assignments

RES0, [31:25]

RES0 Reserved

RST, [24]

Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected:

1 Reset enabled. Multiple matches can occur.

RES0, [23:20]

RES0 Reserved

ARC, [19:16]

Selects one or more address range comparators for single-shot control.

One bit is provided for each implemented address range comparator.

RES0, [15:8]

RES0 Reserved

SAC, [7:0]

Selects one or more single address comparators for single-shot control.

One bit is provided for each implemented single address comparator.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSSCCR0 can be accessed through the external debug interface, offset `0x280`.

D10.67 TRCSSCSR0, Single-Shot Comparator Status Register 0

The TRCSSCSR0 indicates the status of the single-shot comparator. TRCSSCSR0 is sensitive to instruction addresses.

Bit field descriptions

The TRCSSCSR0 is a 32-bit register

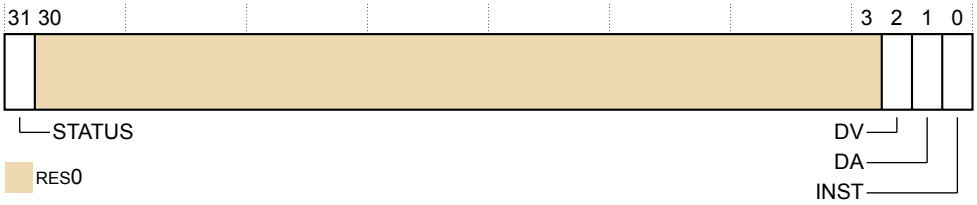

Figure D10-63 TRCSSCSR0 bit assignments

STATUS, [31]

Single-shot status. This indicates whether any of the selected comparators have matched:

`0` Match has not occurred.

`1` Match has occurred at least once.

When programming the ETM trace unit, if TRCSSCCRn.RST is b0, the STATUS bit must be explicitly written to 0 to enable this single-shot comparator control.

RES0, [30:3]

RES0 Reserved

DV, [2]

Data value comparator support:

`0` Single-shot data value comparisons not supported

DA, [1]

Data address comparator support:

`0` Single-shot data address comparisons not supported

INST, [0]

Instruction address comparator support:

`1` Single-shot instruction address comparisons supported

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSSCSR0 can be accessed through the external debug interface, offset `0x2A0`.

D10.68 TRCSTATR, Status Register

The TRCSTATR indicates the ETM trace unit status.

Bit field descriptions

The TRCSTATR is a 32-bit register.

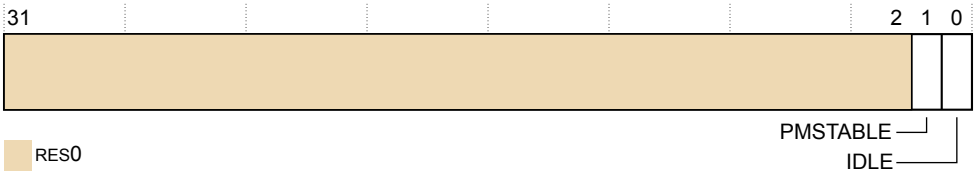

Figure D10-64 TRCSTATR bit assignments

RES0, [31:2]

RES0 Reserved

PMSTABLE, [1]

Indicates whether the ETM trace unit registers are stable and can be read:

`0` The programmers model is not stable.

`1` The programmers model is stable.

IDLE, [0]

Idle status:

`0` The ETM trace unit is not idle.

`1` The ETM trace unit is idle.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSTATR can be accessed through the external debug interface, offset `0x00C`.

D10.69 TRCSYNCPR, Synchronization Period Register

The TRCSYNCPR controls how often periodic trace synchronization requests occur.

Bit field descriptions

The TRCSYNCPR is a 32-bit register.

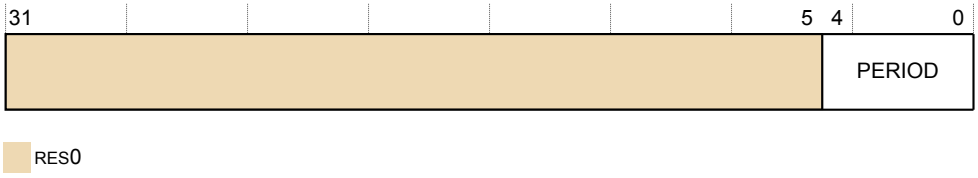

Figure D10-65 TRCSYNCPR bit assignments

RES0, [31:5]

RES0 Reserved

PERIOD, [4:0]

Defines the number of bytes of trace between synchronization requests as a total of the number of bytes generated by both the instruction and data streams. The number of bytes is 2^N where N is the value of this field:

- A value of zero disables these periodic synchronization requests, but does not disable other synchronization requests.
- The minimum value that can be programmed, other than zero, is 8, providing a minimum synchronization period of 256 bytes.
- The maximum value is 20, providing a maximum synchronization period of 2^{20} bytes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCSYNCPR can be accessed through the external debug interface, offset `0x034`.

D10.70 TRCTRACEIDR, Trace ID Register

The TRCTRACEIDR sets the trace ID for instruction trace.

Bit field descriptions

The TRCTRACEIDR is a 32-bit register.

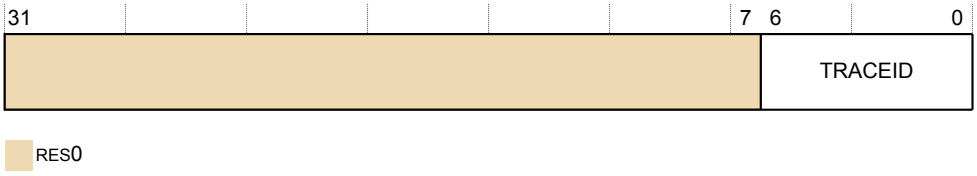

Figure D10-66 TRCTRACEIDR bit Assignments

RES0, [31:7]

RES0 Reserved

TRACEID, [6:0]

Trace ID value. When only instruction tracing is enabled, this provides the trace ID.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCTRACEIDR can be accessed through the external debug interface, offset `0x040`.

D10.71 TRCTSCTLR, Global Timestamp Control Register

The TRCTSCTLR controls the insertion of global timestamps in the trace streams. When the selected event is triggered, the trace unit inserts a global timestamp into the trace streams. The event is selected from one of the Resource Selectors.

Bit field descriptions

The TRCTSCTLR is a 32-bit register.

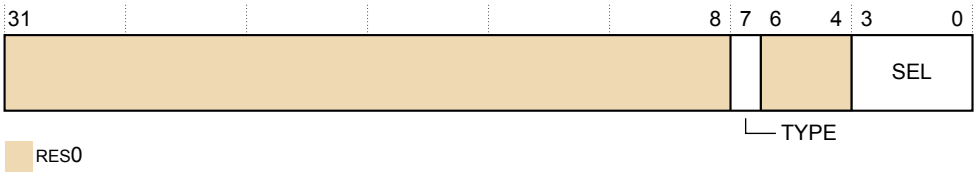

Figure D10-67 TRCTSCTLR bit assignments

RES0, [31:8]

RES0 Reserved

TYPE, [7]

Single or combined resource selector

RES0, [6:4]

RES0 Reserved

SEL, [3:1]

Identifies the resource selector to use

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCTSCTLR can be accessed through the external debug interface, offset `0x030`.

D10.72 TRCVICTLR, ViewInst Main Control Register

The TRCVICTLR controls instruction trace filtering.

Bit field descriptions

The TRCVICTLR is a 32-bit register.

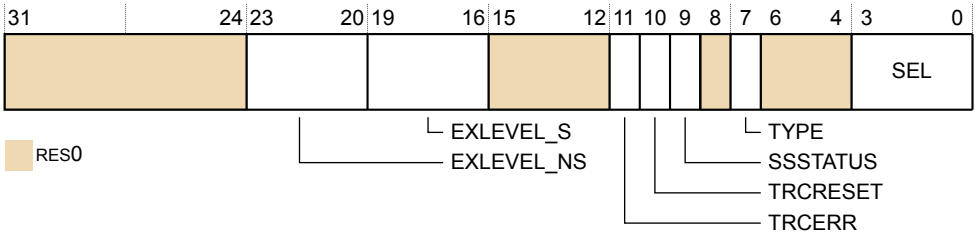

Figure D10-68 TRCVICTLR bit assignments

RES0, [31:24]

RES0 Reserved

EXLEVEL_NS, [23:20]

In Non-secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level:

0 Trace unit generates instruction trace, in Non-secure state, for Exception level *n*.

1 Trace unit does not generate instruction trace, in Non-secure state, for Exception level *n*.

The Exception levels are:

Bit[20] Exception level 0

Bit[21] Exception level 1

Bit[22] Exception level 2

Bit[23] RAZ/WI. Instruction tracing is not implemented for Exception level 3.

EXLEVEL_S, [19:16]

In Secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level:

0 Trace unit generates instruction trace, in Secure state, for Exception level *n*.

1 Trace unit does not generate instruction trace, in Secure state, for Exception level *n*.

The Exception levels are:

Bit[16] Exception level 0

Bit[17] Exception level 1

Bit[18] RAZ/WI. Instruction tracing is not implemented for Exception level 2.

Bit[19] Exception level 3

RES0, [15:12]

RES0 Reserved

TRCERR, [11]

Selects whether a system error exception must always be traced:

`0` System error exception is traced only if the instruction or exception immediately before the system error exception is traced.

`1` System error exception is always traced regardless of the value of ViewInst.

TRCRESET, [10]

Selects whether a reset exception must always be traced:

`0` Reset exception is traced only if the instruction or exception immediately before the reset exception is traced.

`1` Reset exception is always traced regardless of the value of ViewInst.

SSSTATUS, [9]

Indicates the current status of the start/stop logic:

`0` Start/stop logic is in the stopped state.

`1` Start/stop logic is in the started state.

RES0, [8]

RES0 Reserved

TYPE, [7]

Selects the resource type for the viewinst event:

`0` Single selected resource

`1` Boolean combined resource pair

RES0, [6:4]

RES0 Reserved

SEL, [3:0]

Selects the resource number to use for the viewinst event, based on the value of TYPE:

When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVICTLR can be accessed through the external debug interface, offset `0x080`.

D10.73 TRCVIIECTLR, ViewInst Include-Exclude Control Register

The TRCVIIECTLR defines the address range comparators that control the ViewInst Include/Exclude control.

Bit field descriptions

The TRCVIIECTLR is a 32-bit register.

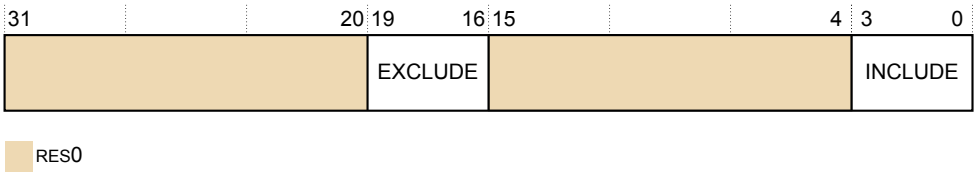

Figure D10-69 TRCVIIECTLR bit assignments

RES0, [31:20]

RES0 Reserved

EXCLUDE, [19:16]

Defines the address range comparators for ViewInst exclude control. One bit is provided for each implemented Address Range Comparator.

RES0, [15:4]

RES0 Reserved

INCLUDE, [3:0]

Defines the address range comparators for ViewInst include control.

Selecting no include comparators indicates that all instructions must be included. The exclude control indicates which ranges must be excluded.

One bit is provided for each implemented Address Range Comparator.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVIIECTLR can be accessed through the external debug interface, offset `0x084`.

D10.74 TRCVISSCTLR, ViewInst Start-Stop Control Register

The TRCVISSCTLR defines the single address comparators that control the ViewInst Start/Stop logic.

Bit field descriptions

The TRCVISSCTLR is a 32-bit register.

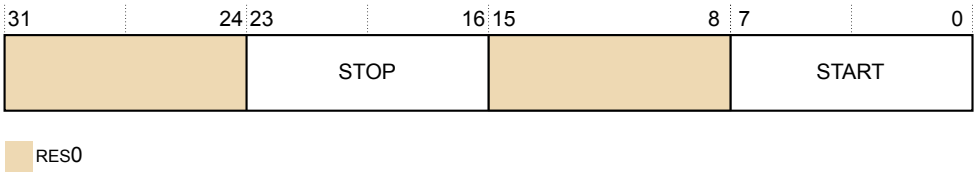

Figure D10-70 TRCVISSCTLR bit assignments

RES0, [31:24]

RES0 Reserved

STOP, [23:16]

Defines the single address comparators to stop trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

RES0, [15:8]

RES0 Reserved

START, [7:0]

Defines the single address comparators to start trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVISSCTLR can be accessed through the external debug interface, offset `0x088`.

D10.75 TRCVMIDCVR0, VMID Comparator Value Register 0

The TRCVMIDCVR0 contains a VMID value.

Bit field descriptions

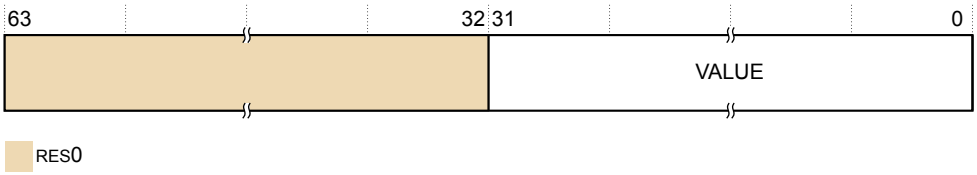

Figure D10-71 TRCVMIDCVR0 bit assignments

RES0, [63:32]

RES0 Reserved

VALUE, [31:0]

The VMID value

The TRCVMIDCVR0 can be accessed through the internal memory-mapped interface and the external debug interface, offset `0x640`.

Usage constraints

Accepts writes only when the trace unit is disabled.

Configurations

Available in all configurations.

D10.76 TRCVMIDCCTLR0, Virtual context identifier Comparator Control Register 0

The TRCVMIDCCTLR0 contains the Virtual machine identifier mask value for the TRCVMIDCVR0 register.

Bit field descriptions

The TRCVMIDCCTLR0 is a 32-bit register.

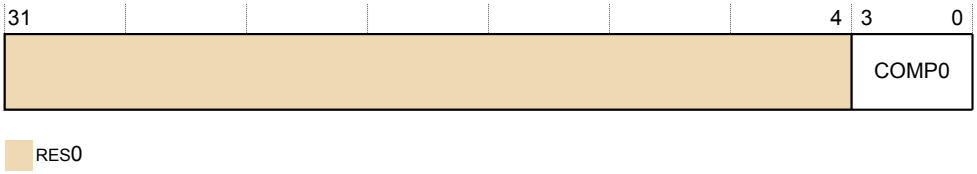

Figure D10-72 TRCVMIDCCTLR0 bit assignments

RES0, [31:4]

RES0 Reserved

COMP0, [3:0]

Controls the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0. When a bit is:

`0` The trace unit includes the relevant byte in TRCVMIDCVR0 when it performs the Virtual context ID comparison.

`1` The trace unit ignores the relevant byte in TRCVMIDCVR0 when it performs the Virtual context ID comparison.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Embedded Trace Macrocell Architecture Specification ETMv4*.

The TRCVMIDCCTLR0 can be accessed through the external debug interface, offset `0x688`.

Chapter D11
SPE registers

This chapter describes the *Statistical Profiling Extension* (SPE) registers.

It contains the following section:

D11.1 SPE register summary

This section summarizes the *Statistical Profiling Extension* (SPE) registers.

The following table lists all of the SPE registers included in the SPE architecture.

Table D11-1 AArch64 debug register summary

Op0	Op1	CRn	CRm	Op2	Name	Type	Reset	Description
3	0	c9	c9	0	PMSCR_EL1	RW	UNK	Statistical Profiling Control Register EL1
3	4	c9	c9	0	PMSCR_EL2	RW	UNK	Statistical Profiling Control Register EL2
3	5	c9	c9	0	PMSCR_EL12	RW	UNK	Alias of the PMSCR_EL1 register, available in EL2
3	0	c9	c9	2	PMSICR_EL1	RW	UNK	Sampling Interval Counter Register
3	0	c9	c9	3	PMSIRR_EL1	RW	UNK	Sampling Interval Reload Register
3	0	c9	c9	5	PMSEVFR_EL1	RW	UNK	Sampling Event Filter Register
3	0	c9	c9	6	PMSLATFR_EL1	RW	UNK	Sampling Latency Filter Register
3	0	c9	c10	1	PMBPTR_EL1	RW	UNK	Profiling Buffer Write Pointer Register
3	0	c9	c10	0	PMBLIMITR_EL1	RW	00000000	Profiling Buffer Limit Address Register
3	0	c9	c10	3	PMBSR_EL1	RW	UNK	Profiling Buffer Status/syndrome Register
3	0	c9	c9	4	PMSFCR_EL1	RW	UNK	Sampling Filter Control Register
3	0	c9	c10	7	PMBIDR_EL1	RO	00000026	Profiling Buffer ID Register
3	0	c9	c9	7	PMSIDR_EL1	RO	00026497	Sampling Profiling ID Register

Part E
Appendices

Appendix A
Cortex-A78C Core AArch32 UNPREDICTABLE behaviors

This appendix describes the cases in which the Cortex-A78C core implementation diverges from the preferred behavior described in Armv8 AArch32 UNPREDICTABLE behaviors.

It contains the following sections:

A.1 Use of R15 by Instruction

If the use of R15 as a base register for a load or store is UNPREDICTABLE, the value used by the load or store using R15 as a base register is the *Program Counter* (PC) with its usual offset and, in the case of T32 instructions, with the forced word alignment. In this case, if the instruction specifies Writeback, then the load or store is performed without Writeback.

The Cortex-A78C core does not implement a *Read 0* or *Ignore Write* policy on UNPREDICTABLE use of R15 by instruction. Instead, the Cortex-A78C core takes an UNDEFINED exception trap.

A.2 Load/Store accesses crossing page boundaries

The Cortex-A78C core implements a set of behaviors for load or store accesses that cross page boundaries.

Crossing a page boundary with different memory types or shareability attributes

The *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, states that a memory access from a load or store instruction that crosses a page boundary to a memory location that has a different memory type or shareability attribute results in CONSTRAINED UNPREDICTABLE behavior.

Crossing a 4KB boundary with a Device access

The *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*, states that a memory access from a load or store instruction to Device memory that crosses a 4KB boundary results in CONSTRAINED UNPREDICTABLE behavior.

Implementation (for both page boundary specifications)

For an access that crosses a page boundary, the Cortex-A78C core implements the following behaviors:

- Store crossing a page boundary:
  - No alignment fault.
  - The access is split into two stores.
  - Each store uses the memory type and shareability attributes associated with its own address.
- Load crossing a page boundary (Device to Device and Normal to Normal):
  - No alignment fault.
  - The access is split into two loads.
  - Each load uses the memory type and shareability attributes associated with its own address.
- Load crossing a page boundary (Device to Normal and Normal to Device):
  - The instruction will generate an alignment fault.

A.3 Armv8 Debug UNPREDICTABLE behaviors

This section describes the behavior that the Cortex-A78C core implements when:

- A topic has multiple options.
- The behavior differs from either or both of the Options and Preferences behaviors.

Note

This section does not describe the behavior when a topic only has a single option and the core implements the preferred behavior.

Table A-1 Armv8 Debug UNPREDICTABLE behaviors

Scenario	Behavior
A32 BKPT instruction with condition code not AL	The core implements the following preferred option: • Executed unconditionally.
Address match breakpoint match only on second halfword of an instruction	The core generates a breakpoint on the instruction if CPSR.IL=0. In the case of CPSR.IL=1, the core does not generate a breakpoint exception.
Address matching breakpoint on A32 instruction with DBGBCRn.BAS=1100	The core implements the following option: • Does match if CPSR.IL=0.
Address match breakpoint match on T32 instruction at DBGBCRn+2 with DBGBCRn.BAS=1111	The core implements the following option: • Does match.
Link to non-existent breakpoint or breakpoint that is not context-aware	The core implements the following option: • No Breakpoint or Watchpoint debug event is generated, and the LBN field of the *linker* reads UNKNOWN.
DBGWCRn_EL1.MASK!=00000 and DBGWCRn_EL1.BAS!=11111111	The core behaves as indicated in the sole Preference: • DBGWCRn_EL1.BAS is IGNORED and treated as if `0x11111111`.
Address match breakpoint with DBGBCRn_EL1.BAS=0000	The core implements the following option: • As if disabled.
DBGWCRn_EL1.BAS specifies a non-contiguous set of bytes within a double-word	The core implements the following option: • A Watchpoint debug event is generated for each byte.
A32 HLT instruction with condition code not AL	The core implements the following option: • Executed unconditionally.
Execute instruction at a given EL when the corresponding EDECCR bit is 1 and Halting is allowed	The core behaves as follows: • Generates debug event and Halt no later than the instruction following the next *Context Synchronization operation* (CSO) excluding ISB instruction.
H > N or H = 0 at Non-secure EL1 and EL0, including value read from PMCR_EL0.N	The core implements: • A simple implementation where all of HPMN[4:0] are implemented, and In Non-secure EL1 and EL0: — If H > N then M = N. — If H = 0 then M = 0.
H > N or H = 0: value read back in MDCR_EL2.HPMN	The core implements: • A simple implementation where all of HPMN[4:0] are implemented and for reads of MDCR_EL2.HPMN, return H.

Table A-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
P ≥ M and P ≠ 31: reads and writes of PM XEVTYPER_EL0 and PMXEVCNTR_EL0	The core implements: • A simple implementation where all of SEL[4:0] are implemented, and if P ≥ M and P ≠ 31 then the register is RES0.
P ≥ M and P ≠ 31: value read in PMSELR_EL0.SEL	The core implements: • A simple implementation where all of SEL[4:0] are implemented, and if P ≥ M and P ≠ 31 then the register is RES0.
P = 31: reads and writes of PMXEVCNTR_EL0	The core implements: • RES0.
n ≥ M: Direct access to PMEVCNTRn_EL0 and PMEVTYPERn_EL0	The core implements: • If n ≥ N, then the instruction is UNALLOCATED. • Otherwise if n ≥ M, then the register is RES0.
Exiting Debug state while instruction issued through EDITR is in flight	The core implements the following option: • The instruction completes in Debug state before executing the restart.
Using memory-access mode with a non-word-aligned address	The core behaves as indicated in the sole Preference: • Does unaligned accesses, faulting if these are not permitted for the memory type.
Access to memory-mapped registers mapped to Normal memory	The core behaves as indicated in the sole Preference: • The access is generated, and accesses might be repeated, gathered, split or resized, in accordance with the rules for Normal memory, meaning the effect is UNPREDICTABLE.
Not word-sized accesses or (AArch64 only) doubleword-sized accesses >	The core behaves as indicated in the sole Preference: • Reads occur and return UNKNOWN data. • Writes set the accessed register(s) to UNKNOWN.
External debug write to register that is being reset	The core behaves as indicated in the sole Preference: • Takes reset value.

Table A-1 Armv8 Debug UNPREDICTABLE behaviors (continued)

Scenario	Behavior
Accessing reserved debug registers	The core deviates from preferred behavior because the hardware cost to decode some of these addresses in debug power domain is significantly high. The actual behavior is: 1. For reserved debug registers in the address range `0x000-0xCFC` and Performance Monitors registers in the address range `0x000`, the response is either CONSTRAINED UNPREDICTABLE Error or RES0 when any of the following errors occurs: **Off** The core power domain is either completely off or in a low-power state where the core power domain registers cannot be accessed. **DLK** `DoubleLockStatus()` is TRUE and OS double-lock is locked (EDPRSR.DLK is 1). **OSLK** OS lock is locked (OSLSR_EL1.OSLK is 1). 2. For reserved debug registers in the address ranges `0x400-0x4FC` and `0x800-0x8FC`, the response is CONSTRAINED UNPREDICTABLE Error or RES0 when the conditions in *1* do not apply and the following error occurs: **EDAD** `AllowExternalDebugAccess()` is FALSE. External debug access is disabled. 3. For reserved Performance Monitor registers in the address ranges `0x000-0x0FC` and `0x400-0x47C`, the response is either CONSTRAINED UNPREDICTABLE Error, or RES0 when the conditions in *1* and *2* do not apply, and the following error occurs: **EPMAD** `AllowExternalPMUAccess()` is FALSE. External Performance Monitors access is disabled.
Clearing the *clear-after-read* EDPRSR bits when Core power domain is on, and `DoubleLockStatus()` is TRUE	The core behaves as indicated in the sole Preference: • Bits are not cleared to zero.

A.4 Other UNPREDICTABLE behaviors

This section describes other UNPREDICTABLE behaviors.

Table A-2 Other UNPREDICTABLE behaviors

Scenario	Description
CSSELR indicates a cache that is not implemented.	If CSSELR indicates a cache that is not implemented, then on a read of the CCSIDR the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following: • The CCSIDR read is treated as `NOP`. • The CCSIDR read is UNDEFINED. • The CCSIDR read returns an UNKNOWN value (preferred).
HDCR.HPMN is set to 0, or to a value larger than PMCR.N.	If HDCR.HPMN is set to 0, or to a value larger than PMCR.N, then the behavior in Non-secure EL0 and EL1 is CONSTRAINED UNPREDICTABLE, and one of the following must happen: • The number of counters accessible is an UNKNOWN non-zero value less than PMCR.N. • There is no access to any counters. For reads of HDCR.HPMN by EL2 or higher, if this field is set to 0 or to a value larger than PMCR.N, the core must return a CONSTRAINED UNPREDICTABLE value that is one of: • PMCR.N. • The value that was written to HDCR.HPMN. • (The value that was written to HDCR.HPMN) modulo 2h, where h is the smallest number of bits required for a value in the range 0 to PMCR.N.
CRC32 or CRC32C instruction with `size==64`.	On read of the instruction, the behavior is CONSTRAINED UNPREDICTABLE, and the instruction executes with the additional decode: `size==32`.
CRC32 or CRC32C instruction with `cond!=1110` in the A1 encoding.	The core implements the following option: • Executed unconditionally.

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

- *B.1 Revisions* on page Appx-B-704.

B.1 Revisions

This appendix describes the technical changes between released issues of this book.

Table B-1 Issue 0001-01

Change	Location
First release for r0p1	-